Grupo de Trabajo en Red
Comentarios: 2060
Obsoletos: 1730
Categoría: En proceso de estandarización

M.Crispin
Universidad de Washington

Diciembre 1996

PROTOCOLO DE ACCESO A MENSAJES DE INTERNET - VERSION 4rev1

Status de este memorándum

Este documento especifica un protocolo de Internet en vías de estandarización, y es susceptible de recibir críticas y sugerencias para su mejora. Le remito a la edición actual de "Estándares de Protocolos Oficiales de Internet" para conocer el estado de estandarización y status de este protocolo. La distribución de este memorándum es ilimitada.

Resumen

El protocolo de acceso a mensajes de Internet, versión 4rev1 (IMAP4rev1) permite a un cliente acceder y manipular los mensajes de correo electrónico contenidos en un servidor. IMAP4rev1 nos permite manipular las carpetas remotas contenedoras de mensajes, llamadas comúnmente "buzones", con una funcionalidad equivalente a la que obtenemos con los contenedores de mensajes locales. IMAP4rev1 también ofrece la posibilidad de que un cliente "offline", establezca una sincronización con el servidor.

IMAP4rev1 ofrece además, operaciones para crear, borrar, y renombrar estos buzones; comprobación de nuevos mensajes; borrado permanente de mensajes; establecimiento y borrado de banderas; procesado [RFC-822] y [MIME-IMB];búsqueda genérica; y recuperación selectiva de atributos de mensaje, textos, y porciones de texto. El acceso a mensajes a través de IMAP4rev1 se realiza a través de números. Estos números pueden ser, una secuencia de números, o bien identificadores únicos.

IMAP4rev1 da soporte a un único servidor. Para más información acerca de configuraciones pensadas para múltiples servidores le remitimos a [ACAP].

IMAP4rev1 no especifica un medio de distribución de correo; esta función es desempeñada por un protocolo de transferencia de correo como [SMTP].

IMAP4rev1 se ha diseñado para mantener la compatibilidad con los protocolos anteriores partiendo desde [IMAP2] y los protocolos IMAP2bisno publicados. A lo largo del tiempo de vida de IMAP4rev1, algunos aspectos del protocolo inicial han quedado obsoletos. Estos

aspectos, como pueden ser, comandos, respuestas, y formatos de datos con los que trataba IMAP4rev1 en sus versiones más tempranas se describen en [IMAP-OBSOLETE].

Todo lo relacionado con compatibilidades con IMAP2bis, la versión más común del protocolo más primitivo, se discute en [IMAP-COMPAT]. Una discusión más a fondo acerca de los problemas de compatibilidad con variantes poco usadas de IMAP2 se encuentra en [IMAP-HISTORICAL]; este documento es en esencia, de interés histórico.

Tabla de contenidos

Especificación del protocolo IMAP4rev1

1. Cómo leer este documento

1.1. Estructura de este documento

Este documento está escrito desde la óptica tanto del implementador

del cliente IMAP4rev1 como del servidor IMAPrev1. Más allá del primer contacto tomado en la sección 2, este documento no está dirigido a aquella persona que pretenda comprender el modo de operar de este protocolo. El contenido que va desde la sección 3 hasta la sección 5 expone el contexto general y las definiciones con las que IMAP4rev1 trata.

Las secciones 6, 7, y 9 describen los comandos, respuestas, y sintaxis respectivamente de IMAP. La relación existente entre ellos es tal, que es prácticamente imposible comprender cualquiera de ellos por separado. En concreto, no intente comprender la sintaxis de los comandos a partir de la sección de comandos, sino que inténtelo en la sección de sintaxis formal.

1.2. Convenciones utilizadasen este documento

Por ejemplo, "C:" y "S:" indica respectivamente, las líneas enviadas por el cliente y las enviadas por el servidor.

Los siguientes términos se utilizan en el documento para hacer patentes los requerimientos de esta especificación.

1) DEBE, o el adjetivo REQUERIDO, significa que la definición es un requerimiento imprescindible en la especificación.

2) NO DEBE, la definición está absolutamente prohibida por la especificación.

3) DEBERIA significa que es posible que existan motivos razonables en determinadas circunstancias para ignorar un elemento en particular, pero todas las consecuencias que se derivan de ello deben ser conocidas y sopesadas cuidadosamente antes de actuar de manera diferente.

4) NO DEBERIA significa que es posible que existan motivos razonables en determinadas circunstancias en las cuales el comportamiento concreto sea aceptable o incluso útil, pero las posibles consecuencias DEBERIAN ser conocidas y en su caso sopesadas antes de actuar conforme a un dictado descrito con esta etiqueta.

5) ES POSIBLE, o el adjetivo OPCIONAL, significa que un elemento es de hecho, opcional. Un vendedor puede incluir dicho elemento en un momento dado por los requerimientos del mercado, o también porque cree que potencia el producto, mientras que otro vendedor puede omitir el mismo elemento. Un desarrollo que no incluya una opción en particular DEBE estar preparado para operar con otro desarrollo que sí lo incluya.

"Poder" se utiliza en lugar de "Es posible" cuando nos referimos a una situación o circunstancia que es factible que pueda darse, en contraposición a una característica opcional del protocolo.

"Usuario" se utiliza para hacer referencia al usuario en cuestión, mientras que "cliente" hace referencia al software utilizado por dicho usuario.

"Conexión" hace referencia a la secuencia total de comunicaciones ocurridas desde el establecimiento inicial de la conexión de red hasta su finalización. "Sesión" se refiere a la secuencia de comunicaciones Cliente/servidor a partir del momento en que se selecciona un buzón (comando SELECT o EXAMINE) hasta el momento en que esta selección termina. (SELECT o EXAMINE sobre otro buzón, comando CLOSE, o finalización de la conexión).

Se utilizan caracteres de 7 bit US-ASCII salvo que se especifique otro formato diferente. Para utilizar otro conjunto de caracteres se indica usando un "CHARSET", como se describe en [MIME-IMT] y se define en [CHARSET]. Los CHARSETs tienen una semántica adicional importante además de dar la definición de un conjunto de caracteres; para más información le remito a los documentos anteriores.

2. Primer contacto con el protocolo

2.1. Nivel de enlace

El protocolo IMAP4rev1 confía en la existencia de un flujo de datos estable, como los suministrados por TCP. Cuando utilizamos TCP, un servidor IMAP4rev1 escucha el puerto 143.

2.2. Comandos y respuestas

Una conexión IMAP4rev1 consiste en el establecimiento de una conexión de red cliente/servidor, un reconocimiento inicial por parte del servidor, y una serie de comunicaciones cliente/servidor. Estas comunicaciones consisten en un comando de cliente, datos del servidor, y una respuesta por parte del servidor de transferencia completada.

La comunicación entre cliente y servidor se realiza en forma de líneas; es decir, cadenas que terminan con un CTRLF. El receptor de protocolo del cliente o servidor IMAP4rev1 lleva a cabo o la lectura de una línea, o bien la lectura de una secuencia de octetos con un contador conocido seguido de una línea.

2.2.1. Transmisor de protocolo cliente y Receptor de protocolo servidor

El comando de cliente da comienzo a una operación. A cada comando de cliente se le asigna un prefijo consistente en un identificador (normalmente una cadena corta de caracteres alfanuméricos, p.ej. A0001, A0002, etc.) llamado "etiqueta". El cliente asigna a cada comando una etiqueta diferente.

Hay dos casos, en los cuales una línea de cliente no representa un comando completo. En primer lugar, un argumento del comando está dotado de un número de octetos (ver la descripción de literal en Cadena en Formato de Datos); en segundo lugar, los argumentos del comando requieren una realimentación por parte del servidor (ver el comando AUTHENTICATE). En ambos casos, el servidor envía una respuesta de solicitud de continuación del comando si este está preparado para recibir los octetos (si es lo adecuado) y el recordatorio del comando.
Esta respuesta viene precedida del token"+".

> Nota: Si en su caso, el servidor detecta un error en el comando, envía una respuesta de transacción errónea con la etiqueta que corresponde con el comando (como se describe abajo) para rechazar el comando, y con ello evita que el cliente envíe más información del comando.
>
> Es posible que el servidor envíe una respuesta de transacción completa para otros comandos (si hay varios comandos en progreso), o datos no etiquetados. En cualquier caso, la solicitud de continuación de comando está todavía pendiente; el cliente actúa de manera adecuada a la respuesta, y lee otra respuesta del servidor.En todos los casos, el cliente DEBE enviar un comando completo (incluyendo la recepción de todas las respuestas de solicitud de continuación de comando y las continuaciones

de comando para dicho comando) antes de inicializar un nuevo comando.

El receptor de protocolo de un servidor IMAP4rev1 lee una línea de comando desde el cliente, analiza el comando y sus argumentos, y transmite los datos del servidor y una respuesta de transacción completa de comando.

2.2.2. Transmisor de protocolo servidor y Receptor de protocolo cliente

Los datos transmitidos por el servidor hacia el cliente y las respuestas de estado que no indiquen el procesado completo del comando van precedidas del token "*", y se denominan respuestas no etiquetadas.

Los datos del servidor ES POSIBLE que sean enviados como resultado de un comando del cliente, o ES POSIBLE que sean enviados de forma unilateral por el servidor. No hay diferencia sintáctica entre los datos enviados como resultado de un comando del cliente y los enviados de forma unilateral por el servidor.

La respuesta de transacción completa por parte del servidor indica el éxito o fracaso de la operación. Esta es etiquetada con la misma etiqueta con la que el comando del cliente comenzó la operación. Por tanto, si hay más de un comando en progreso, la etiqueta de una respuesta de transacción completa de un servidor identifica al comando al cual corresponde la respuesta. Hay tres posibles respuestas de transacción completa por parte del servidor: OK (indica éxito), NO (indica fracaso), o BAD (indica un error de protocolo tal como comando no reconocido o error de sintaxis para el comando).

El receptor de protocolo cliente IMAP4rev1 lee una línea de la respuesta enviada por el servidor. Después, lleva a cabo una acción en función de dicha respuesta basándose en el primer token que constituya la misma, esta puede ser una etiqueta, un "*" o un "+".

Un cliente DEBE estar preparado para recibir cualquier número de respuestas por parte del servidor en cualquier momento. Esto incluye datos del servidor que no fueron solicitados. Estos datos DEBERIAN ser almacenados, para que el cliente pudiera hacer una referencia a la copia ya almacenada de esos datos, evitando así enviar un comando al servidor para solicitar dichos datos. Ciertos datos enviados por el servidor DEBEN ser almacenados.

Este asunto se discute con más detalle en la sección Respuestas de Servidor.

2.3. Atributos del Mensaje.

Además del texto del mensaje, cada uno de los mensajes posee

una serie de atributos. Estos atributos se pueden recuperar individualmente o en conjunción con otros atributos o textos

del mensaje.

2.3.1. Números del Mensaje.

Para tener acceso a los mensajes en IMAP4rev1 se utilizan uno de los siguientes números; el identificador único y el número de secuencia del mensaje.

2.3.1.1. Atributo de mensaje Identificador Único (UID)

Asignamos a cada mensaje un valor de 32 bit, junto con el valor de vigencia de identificador único obtenemos un valor de 64 bit, con lo cual garantizamos que hacemos referencia al mensaje en concreto y no a cualquier otro presente en el buzón. Los identificadores únicos se asignan en el buzón en orden ascendente; de forma que cualquier mensaje que llegue al buzón recibirá un UID mayor que el identificador asignado al mensaje recibido justo antes que él.

Al contrario que los números de secuencia de mensaje, los identificadores únicos no son necesariamente contiguos. Los identificadores únicos también permanecen a lo largo de sesiones diferentes. Esto permite al cliente resincronizar su estado partiendo desde una sesión previa con el servidor. (p.ej. clientes "offline" o desconectados); esto se discute más ampliamente en [IMAP-DISC].

Todo buzón lleva asociado un valor de vigencia de identificador único, el cual se envía en un código de respuesta UIDVALIDITY dentro de una respuesta de tipo OK no etiquetada en tiempo de selección del buzón. Si algún identificador único de una sesión anterior pierde su vigencia en la sesión actual, el valor de vigencia de identificador único DEBE ser mayor que el usado en la sesión anterior.

Nota: los identificadores únicos DEBEN seguir un orden ascendente en todo momento en la estructura del buzón. Si el almacén físico de mensajes es reordenado por un agente no-IMAP, es preciso que los identificadores únicos del buzón sean regenerados, ya que los primeros identificadores únicos no mantendrán el orden ascendente como resultado de la reorganización. Otra situación en la cual se puede producir una reorganización del buzón es aquella en la que el buzón no disponga de un mecanismo para almacenar identificadores únicos. Aunque esta

especificación reconoce que esto pueda darse en ciertos entornos de servidor, se RECOMIENDA ENCARECIDAMENTE que el almacén de mensajes implemente técnicas que eviten este problema.

Otro causa de no persistencia ocurre si se elimina el buzón y se crea posteriormente uno nuevo con el mismo nombre. Puesto que el nombre es el mismo, puede suceder que el cliente no sepa que dispone de un nuevo buzón a menos que la vigencia del identificador único sea diferente. Una buena práctica consiste en asignar al valor de vigencia de identificador único una representación en 32 bit de la hora/fecha de creación de dicho buzón. Es igualmente correcto, usar una constante como 1, pero únicamente si está garantizado que estos identificadores sean únicos, incluso en el caso de que se borre el buzón y se vuelva a crear uno nuevo con el mismo nombre en una fecha posterior.

El identificador de usuario de un mensaje NO DEBE cambiarse a lo largo de la sesión, y NO DEBERIA ser cambiado entre sesiones. Sin embargo, si no es posible preservar el identificador único de un mensaje en una sesión posterior, cada sesión posterior DEBE tener un nuevo valor de vigencia de identificador único que sea mayor que cualquiera de los ya usados previamente.

2.3.1.2. Atributo de mensaje Número de secuencia de mensaje

Posición relativa desde 1 al número de mensajes en el buzón. Esta posición DEBE estar ordenada ascendentemente por identificador único. Conforme se añadan nuevos mensajes, se le asignará a cada uno un número de secuencia de mensaje que sea una unidad mayor que el número total de mensajes presentes previamente en el buzón.

Los números de secuencia de mensaje pueden ser reasignados a lo largo de la sesión. Por ejemplo, cuando se borra definitivamente un mensaje del buzón (tachado), el número de secuencia de mensaje para los mensajes posteriores se ve decrementado. De la misma manera, un mensaje nuevo puede recibir un número de secuencia de mensaje asignado a otro mensaje antes de su eliminación.

Además de tener acceso a los mensajes por su posición relativa en el buzón, los números de secuencia de mensaje pueden ser usados en cálculos matemáticos. Por ejemplo, si

se recibe un "EXISTS 11" no etiquetado, y anteriormente recibimos un "8 EXISTS" no etiquetado, hemos recibido tres nuevos mensajes con números de secuencia de mensaje 9,10 y 11 respectivamente. Otro ejemplo; si el mensaje 287 almacenado en un buzón contenedor de 523 mensajes tiene un identificador 12345, hay exactamente 286 mensajes poseen un UID menor y 236 mensajes que tienen un UID mayor.

2.3.2. Atributo de mensaje Banderas

Lista de cero o más tokens dotados de nombre asociados con el mensaje. Una bandera se establece añadiéndola a dicha lista, y se elimina borrándola de la misma. Hay dos tipos de banderas en IMPA4rev1. Una bandera sea cual sea su tipo puede ser permanente o de sesión única.

Una bandera de sistema es un nombre de bandera que viene predefinida en esta especificación. Todas las banderas de sistema vienen precedidas de "\". Ciertas banderas de sistema (\Deleted y \Seen) tienen una semántica especial descrita en otra sección. Las banderas de sistema definidas actualmente son:

\Seen	El mensaje ha sido leído
\Answered	El mensaje ha sido respondido
\Flagged	El mensaje está "reseñado" por merecer una atención especial o urgente
\Deleted	El mensaje está "borrado" para ser eliminado posteriormente mediante EXPUNGE
\Draft	El mensaje está inacabado
\Recent	El mensaje ha llegado recientemente al buzón.

Esta sesión es la primera en la que se notificará la llegada de este mensaje, en sesiones posteriores este flag será deshabilitado para este mensaje en concreto. Este flag no puede ser modificado por el cliente.

Si no es posible saber si esta sesión es la primera sesión en la que se deba notificar la llegada del mensaje, entonces este debería ser considerado como reciente.

Si varias conexiones han seleccionado el mismo buzón simultáneamente, será impredecible saber cual de ellas verá los mensajes nuevos con la bandera \Recent activada y cual de ellas la verán desactivada.

Una clave se define por la implementación del servidor. Las claves no comienzan con "\". Los servidores ES POSIBLE que permitan al cliente definir nuevas claves en el buzón. (Ver la descripción del código de respuesta PERMANENTFLAGS para más información).

Una bandera puede ser permanente o de sesión única partiendo de una base por-bandera. Las banderas permanentes son aquellas que el cliente puede añadir o eliminar de las banderas del mensaje de forma permanente; es decir, las sesiones posteriores serán conscientes de cualquier cambio en estas banderas permanentes. Los cambios realizados sobre las banderas de sesión sólo serán válidos en esa sesión en concreto.

Nota: La bandera de sistema \Recent es un caso especial de una bandera de sesión. \Recent no puede ser utilizado como argumento en un comando STORE, y por tanto no se puede cambiar.

2.3.3. Atributo de mensaje Fecha interna

La fecha y hora interna del mensaje en el servidor. Esta no es la fecha y hora de la cabecera [RFC-822], sino que es una fecha y una hora que refleja el momento de recepción del mensaje. En el caso de mensajes distribuidos vía [SMTP], DEBERIA ser la fecha y hora de distribución final del mensaje como se define por [SMTP]. En el caso de los mensajes enviados mediante el comando COPY de IMAP4rev1, esta DEBERIA ser la fecha y hora interna del mensaje origen. En el caso de ser enviado mediante el comando APPEND de IMAPrev1, esta DEBERIA ser la fecha y hora especificada en la descripción del comando APPEND. En cualquier otro caso está definida por la implementación.

2.3.4. Atributo Tamaño de mensaje [RFC-822]

Número de octetos del mensaje, como se expresa en el formato [RFC-822].

2.3.5. Atributo de mensaje Estructura del envoltorio

Una representación procesada acerca de la información de envoltorio [RFC-822] (no confundir con un envoltorio [SMTP]) del mensaje.

2.3.6. Atributo de mensaje Estructura del cuerpo

Una representación procesada de la información sobre la estructura del cuerpo del mensaje [MIME-IMB].

2.4. Textos del mensaje

Además de ser capaz de recuperar todo el texto [RFC-822] de un mensaje, IMAP4rev1 permite la recuperación de fragmentos del texto completo del mensaje. En concreto, es posible recuperar la cabecera del mensaje [RFC-822], el cuerpo del mensaje [RFC-822], una sección del cuerpo [MIME-IMB], o una cabecera [MIME-IMB].

3. Diagrama de estado y de flujo

Un servidor IMAP4rev1 puede encontrarse en cuatro estados. La mayoría de los comandos son válidos únicamente en ciertos estados. Es un error de protocolo de parte del cliente intentar ejecutar un comando mientras este no se encuentra en el estado apropiado. En este caso, un servidor responderá con un resultado de fin de procesado de comando BAD o NO (dependiendo de la implementación del servidor).

3.1. Estado no autentificado

En este estado, el cliente DEBE suministrar credenciales de autentificación antes de que la mayoría de los comandos puedan ser aceptados. Este estado se alcanza cuando comienza la conexión salvo que la conexión haya sido pre-autentificada.

3.2. Estado autentificado

En este estado, el cliente está autentificado y DEBE seleccionar un buzón antes de que se le permita utilizar cualquier comando que afecte a los mensajes contenidos en el mismo. Este estado se alcanza cuando comienza una conexión pre-autentificada, cuando se suministran credenciales de autentificación válidas, o tras un error debido a la selección de un buzón.

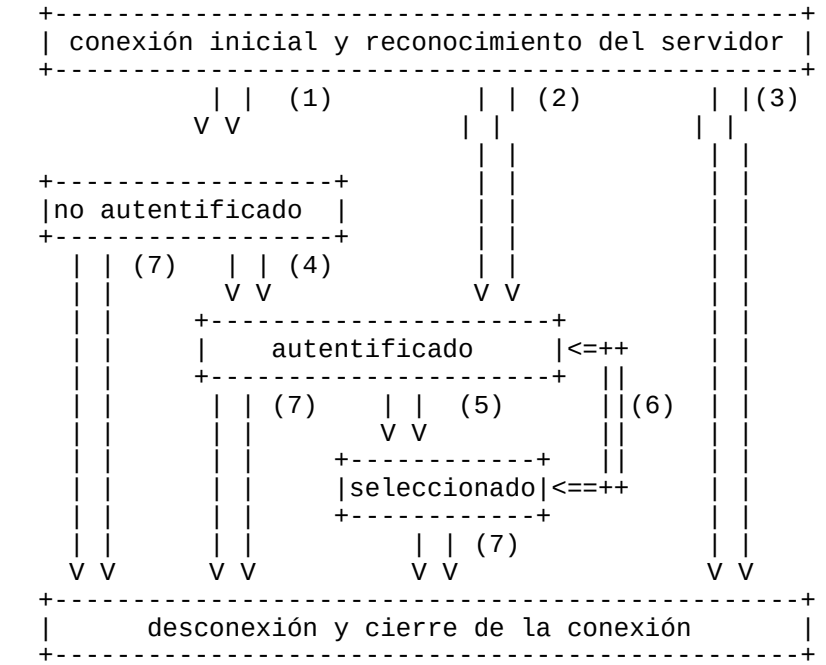
3.3. Estado seleccionado

En este estado, se ha seleccionado un buzón al que acceder.

Este estado se alcanza cuando la selección del buzón es aceptada.

3.4. Estado desconexión

En este estado, la conexión está en proceso de finalización, y el servidor cerrará la conexión. Este estado puede alcanzarse como resultado de la petición del cliente o por decisión unilateral del servidor.



(1) Conexión sin pre-autentificación (reconocimiento OK)
(2) Conexión pre-autentificada (reconocimiento PREAUTH)
(3) Conexión rechazada (reconocimiento BYE)
(4) LOGIN aceptado o comando AUTHENTICATE
(5) SELECT aceptado, o comando EXAMINE
(6) Comando CLOSE, o SELECT rechazado o comando EXAMINE

(7) Comando LOGOUT, apagado del servidor, o conexión cerrada

4. Formatos de datos

IMAP4rev1 utiliza comandos y respuestas en formato texto. Los datos en IMAP4rev1 pueden encontrarse en una de estas cuatro formas: átomo, número, cadena, lista entre paréntesis, o NIL.

4.1 Átomo

Un átomo está constituido de uno o más caracteres no especiales.

4.2. Número

Un número está constituido de uno o más caracteres numéricos,

y representa un valor numérico.

4.3. Cadena

Una cadena puede encontrarse en dos formatos: literal o cadena entre comillas. El literal es la forma genérica de una cadena. La cadena entre comillas es una alternativa que evita la sobrecarga provocada por el procesamiento realizado sobre un literal, como contrapartida tenemos una limitación en cuanto al número de caracteres que pueden utilizarse en una cadena entre comillas.

Un literal es una secuencia de cero o más octetos (incluyendo CR y LF), precedida de un prefijo que encierra una cifra de octetos con el siguiente formato: una llave abierta ("{"), el número de octetos, llave cerrada ("}"), y CRLF. En el caso de los octetos transmitidos desde el servidor al cliente, los datos del octeto van directamente tras el CRFL. En el caso de los octetos transmitidos desde el cliente al servidor, el cliente DEBE esperar a recibir por parte del servidor una solicitud de continuación de comando (esto se explicará más adelante) previo al envío de los datos del octeto. (y el recordatorio del comando).

Una cadena entre comillas es una secuencia de cero o más caracteres 7-bit, excluyendo CR y LF, con comillas dobles (<">) al principio y al final.La cadena vacía se representa como "" (una cadena entre comillas con cero caracteres entre comillas dobles) o como {0} seguido de CRLF (un literal con una cifra de octetos igual a cero).

> Nota: Incluso si la cifra de octetos es igual a 0, un cliente que transmita un literal DEBE esperar a recibir una solicitud de continuación de comando.

4.3.1. Cadenas binarias y de 8-bit

El correo en formato binario y texto de 8-bit se transmite a través del uso de la codificación de transferencia de contenido [MIME-IMB]. Para las implementaciones IMAP4rev1 ES POSIBLE transmitir caracteres de 8-bit o multi-octetos incluidos en literales, pero únicamente debería hacerse cuando el CHARSET se conoce y está identificado.

Aunque está definida una codificación para el cuerpo en BINARIO, no se permiten cadenas binarias no codificadas. Una "cadena en binario" es una cadena con caracteres NUL. Las implementaciones DEBEN codificar los datos binarios en una

representación tipo texto tales como BASE64 antes de que los datos sean transmitidos. Una cadena que contenga una cantidad excesiva de caracteres CTL ES POSIBLE que sea considerada como binaria.

4.4. Lista entre paréntesis

Las estructuras de datos se representan mediante "listas entre paréntesis"; una secuencia de elementos de datos, delimitados por un espacio, y encerrados al principio y al final por paréntesis. Una lista entre paréntesis puede contener otras listas entre paréntesis, haciendo uso así de múltiples niveles de paréntesis para constituir anidamiento.

La lista vacía se representa por () -- una lista entre paréntesis sin elementos.

4.5. NIL

El átomo especial "NIL" representa la no existencia de un determinado elemento de datos, el cual viene definido como cadena o como lista entre paréntesis, siendo diferente de la cadena vacía "" o de la lista entre paréntesis vacía ().

5. Consideraciones operacionales

5.1. Dando nombre al buzón

La interpretación de los nombres de buzón es una interpretación dependiente de la implementación. Sin embargo, el nombre de buzón no sensible a mayúsculas-minúsculas denominado INBOX es un nombre especial que representa "el buzón principal para este usuario en este servidor".

5.1.1. Dando nombre a la jerarquía de buzones

Es deseable que los nombres jerárquicos utilizados en los buzones se exporten, los nombres de los buzones DEBEN ser jerárquicos de izquierda a derecha utilizando un único carácter para separar distintos niveles de la jerarquía. Se utiliza un único carácter separador para la jerarquía en todos los niveles de la misma en el ámbito de un mismo nombre.

5.1.2. Convenciones acerca de la denominación del espacio de

nombres de buzón

Por convención, el primer elemento de la jerarquía de cualquier nombre de buzón que comience con "#" identifica el "espacio de nombres" del recordatorio del nombre. Esto hace posible la distinción entre diferentes tipos de almacén de buzones, cada uno de ellos con sus propios espacios de nombre.

Por ejemplo, las implementaciones que ofrecen acceso a los grupos de noticias USENET ES POSIBLE que utilicen el espacio de nombres "#news" para separar el espacio de nombres del grupo de noticias USENET de otros buzones de correo. Por tanto, el grupo de noticias "#news.comp.mail.misc", y el nombre "comp.mail.misc" podrían hacer referencia a objetos diferentes. (p.ej. el buzón de correo privado de un usuario).

5.1.3. Convenciones acerca de la denominación internacional de los buzones

Por convención, los nombres internacionales de buzón vienen especificados por el uso de una versión modificada de la codificación UTF-7 descrita en [UTF-7]. El propósito de estas modificaciones está dirigido a corregir, utilizando UTF-7, los problemas que se explican a continuación:

1) UTF-7 usa el carácter "+" para espaciado; esto entra en conflicto con el uso común de "+" en los nombres de los buzones, en particular con los nombres de grupos de noticias de USENET.

2) La codificación UTF-7 es BASE64, la cual utiliza el carácter "/"; esto entra en conflicto con el uso de "/" como conocido constructor de jerarquías.

3) UTF-7 prohibe el uso decodificado de "\"; ya que entra en conflicto con el uso de "\" como conocido constructor de jerarquías.

4) UTF-7 prohibe el uso decodificado de "~"; ya que entra en conflicto con el uso de "~" en algunos servidores como indicador de directorio home.

5) UTF-7 permite múltiples maneras para representar la misma cadena; en particular, los caracteres imprimibles US-ASCII pueden ser representados en forma codificada.

En el UTF-7 modificado los caracteres US-ASCII imprimibles salvo el "&"representan el mismo carácter; es decir, los caracteres con valores de octeto 0x20-0x25 y 0x27-0x7e. El carácter "&" (0x26) se representa por la secuencia de dos octetos "&-".

Otros caracteres (valores de octeto 0x00-0x1f, 0x7f-0xff, y todos los octetos de 16-bit Unicode) se representan en BASE64 modificado, con una modificación más profunda de [UTF-7], en la que "," se utiliza en lugar de "/". El BASE64 modificado NO DEBE usarse para representar ningún carácter imprimible US-ASCII que pueda representarse por sí mismo.

"&" se utiliza para activar el BASE64 modificado y "-" para volver al US-ASCII, y DEBE terminar en US-ASCII (un nombre que finalice con un octeto 16-bit Unicode DEBE terminar con un "-").

Por ejemplo, aquí tenemos un nombre de buzón de correo que mezcla Inglés, Japonés y texto Chino:
~peter/mail/&ZeVnLIqe-/&U,BTFw-

5.2 Tamaño y actualización del estado de los mensajes en el buzón

En cualquier momento un servidor puede enviar datos que el cliente no solicitó. En algunas ocasiones, este comportamiento

es OBLIGATORIO. Por ejemplo, otros agentes aparte del servidor ES POSIBLE que añadan mensajes en el buzón de correo (p.ej. nueva distribución de correo), cambiar el estado de las banderas del mensaje presente en el buzón de correo. (p.ej. acceso por parte de múltiples agentes al mismo buzón de correo), o incluso eliminar mensajes del buzón de correo. Un servidor DEBE enviar automáticamente actualizaciones acerca del tamaño del buzón si su tamaño varía durante el procesamiento de un comando. Un servidor DEBERIA enviar automáticamente actualizaciones acerca de las banderas del mensaje, sin necesidad de que el cliente tenga que solicitarlo explícitamente. Existen reglas especiales para que el servidor notifique al cliente la eliminación o borrado de mensajes para evitar errores de sincronización; ver la descripción de la respuesta EXPUNGE para más información.

Sean cuales sean las decisiones de implementación que un cliente establezca para la memorización de datos desde el servidor, cualquier implementación de cliente DEBE registrar las actualizaciones del tamaño del buzón de correo. No DEBE asumirse que cualquier comando ejecutado tras la selección

inicial del buzón de correo devolverá el tamaño de dicho buzón.

5.3. Respuesta sin comando en progreso

Las implementaciones de servidor permiten enviar respuestas no etiquetadas (salvo para EXPUNGE) mientras no exista un comando en progreso. Estas implementaciones DEBEN tomar consideraciones acerca del flujo de control. Especialmente, DEBEN o (1) verificar que el tamaño de los datos no excedan el tamaño de ventana de transporte subyacente, o (2) usen escrituras sin bloqueo.

5.4. Contador de desconexión automática

Si un servidor dispone de un contador de inactividad de desconexión automática, este DEBE ser de al menos 30 minutos de duración. La recepción de un comando enviado por el cliente durante este intervalo DEBERIA suponer la inicialización de dicho contador.

5.5 Múltiples comandos en progreso

El cliente ES POSIBLE que envíe cualquier comando sin necesidad de tener que esperar a la respuesta de resultado tras el procesado completo de un comando, sujetos a las reglas de ambigüedad y restricciones de control de flujo del flujo de datos subyacente. De la misma manera, un servidor ES POSIBLE que comience a procesar otro comando antes de que el tiempo de procesado del comando actual termine, sujeto a las reglas de ambigüedad. Sin embargo, cualquier respuesta de solicitud de continuación de comando y continuación de comandos DEBE ser tratado antes de que cualquier comando posterior sea iniciado.

La excepción surgiría si hubiera ambigüedad como resultado de que un comando afectara a los resultados de otros comandos. Los clientes NO DEBEN enviar múltiples comandos sin esperar que pueda darse ambigüedad. Si el servidor detecta una posible ambigüedad, DEBE ejecutar comandos para mantener la integridad en el orden dado por el cliente.

El ejemplo más claro de ambigüedad es cuando un comando afecta

el resultado de otro; por ejemplo, un FETCH sobre las banderas de un mensaje y un STORE sobre las mismas banderas de dicho mensaje.

Una situación de ambigüedad no tan clara ocurre con aquellos

comandos que permiten una respuesta EXPUNGE no etiquetada (comandos diferentes a FETCH, STORE, y SEARCH), ya que una respuesta EXPUNGE no etiquetada puede invalidar números de secuencia de un comando posterior. Esto no supone un problema para los comandos FETCH, STORE, o SEARCH puesto que a los servidores no les está permitido enviar respuestas EXPUNGE mientras está en progreso cualquiera de estos comandos. Por tanto, si el cliente envía un comando que no sea FETCH, STORE o SEARCH, DEBE esperar una respuesta antes deenviar un comando con números de secuencia de mensaje.

Por ejemplo, las siguientes secuencias de comando sin espera no son válidas:

```
FETCH + NOOP + STORE
STORE + COPY + FETCH
COPY + COPY
CHECK + FETCH
```

Los siguientes son ejemplos de secuencias de comando sin espera válidas:

```
FETCH + STORE + SEARCH + CHECK
STORE + COPY + EXPUNGE
```

6. Comandos de cliente

En esta sección se detallan los comandos IMAP4rev1. Los comandos se organizan en función del estado en el cual está permitida su ejecución. Los comandos que puedan utilizarse en varios estados se listan en el estado mínimo permitido. (por ejemplo, los comandos válidos en los estados autentificado y seleccionado se listan entre los comandos permitidos en el estado autentificado).

Los argumentos de comando, identificados por "Argumentos:" en las descripciones de los comandos situadas más abajo, se describen por su función, no por su sintaxis. La sintaxis precisa de los argumentos de cada comando se describe en la sección Sintaxis Formal.

Algunos comandos provocan la devolución de determinadas respuestas por parte del servidor; estas se identifican por "Respuestas:" en la descripciones de los comandos situadas más abajo. Para más información acerca de estas respuestas y sus descripciones diríjase a la sección de Respuestas, y a la sección de Sintaxis Formal en caso de que desee conocer más a fondo la sintaxis de estas respuestas. Es posible que

se transmitan datos del servidor como resultado de la ejecución un comando; por tanto, para comandos que no requieran datos del servidor especifique "no respuestas específicas para este comando" en lugar de especificar

"ninguna".

"Result:" en la descripción de comandos se refiere a las posibles respuestas de estado etiquetadas para un comando, y cualquier otra interpretación especial de estas respuestas de estado.

6.1. Comandos de cliente - Estado cualesquiera

Los siguientes comandos son perfectamente válidos en cualquiera de los estados: CAPABILITY, NOOP, y LOGOUT.

6.1.1.Comando CAPABILITY

Argumentos: ninguno

Respuestas: respuesta no etiquetada REQUERIDA: CAPABILITY

Resultado: OK comando CAPABILITY completado
BAD comando desconocido o argumentos no válidos

El comando CAPABILITY solicita un listado de todas las aptitudes de que dispone el servidor. El servidor DEBE enviar una respuesta CAPABILITY no etiquetada incluyendo entre la lista de aptitudes "IMAP4rev1" antes de la respuesta (etiquetada) OK. Este listado de aptitudes no depende del estado de la conexión o del usuario. Por lo tanto, no es necesario elaborar un comando CAPABILITY más de una vez por conexión.

Un nombre de aptitud que comience con "AUTH=" indica que el servidor dispone de este mecanismo de autentificación en particular. Tales nombres son, por definición, parte de esta especificación. Por ejemplo, la aptitud de autorización para un autentificador experimental llamado blurdybool sería "AUTH=BLURDYBLOOP" y no "XAUTH=BLURDYBLOOP" o "XAUTH=XBLURDYBLOOP".

Otros nombres de aptitud se refieren a extensiones, revisiones, o añadidos a esta especificación. Para más información diríjase a la documentación referente a la respuesta CAPABILITY. No hay aptitudes más allá del

conjunto IMAP4rev1 base definido en esta especificación, que puedan invocar la aptitud sin una acción especifica del cliente que la invoque.

Ver la sección titulada "Comandos de Cliente - Experimental/En Desarrollo" para más información acerca de las aptitudes específicas de cada implementación.

Ejemplo: C: abcd CAPABILITY
S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4
S: abcd OK CAPABILITY completado

6.1.2. Comando NOOP

Argumentos: ninguno

Respuestas: no hay respuestas especificas para este comando

Resultado: OK noop completado
BAD comando desconocido o argumentos no

válidos

El comando NOOP siempre se ejecuta correctamente. No hace nada.

Puesto que un comando puede devolver como dato no etiquetado una actualización de estado, el comando NOOP puede ser utilizado como un centinela permanente para comprobación de nuevos mensajes o de actualizaciones sobre el estado de los mensajes durante un periodo de inactividad. El comando NOOP también se puede utilizar para inicializar cualquier contador de desconexión por inactividad presente en el servidor.

Ejemplo: C: a002 NOOP
S: a002 OK NOOP completada

. . .
C: a047 NOOP
S: * 22 EXPUNGE
S: * 23 EXISTS
S: * 3 RECENT
S: * 14 FETCH (FLAGS (\Seen \Deleted))
S: a047 OK NOOP completada

6.1.3. Comando LOGOUT

Argumentos : ninguno

Respuestas: respuesta no etiquetada OBLIGATORIA : BYE

Resultado: OK logout completado
BAD comando desconocido o argumentos no válidos

El comando LOGOUT informa al servidor de que el cliente ha conseguido conectarse. El servidor DEBE enviar una respuesta no etiquetada BYE antes de la respuesta (etiquetada) OK, y a continuación cerrar la conexión de red.

Ejemplo: C: A023 LOGOUT
S: * BYE IMAP4rev1 servidor cerrando conexión
S: A023 OK LOGOUT completado
(Servidor y cliente cierran la conexión)

6.2. Comandos de cliente - Estado no autentificado

En el estado no autentificado, el comando AUTHENTICATE o LOGIN establece la autentificación y nos introduce en un estado autentificado. El comando AUTHENTICATE ofrece un mecanismo global para una gran variedad de técnicas de autentificación, mientras que el comando LOGIN hace uso del par nombre de usuario tradicional y contraseña de texto plano.

Las implementaciones de servidor ES POSIBLE que permitan un acceso no autentificado a ciertos buzones de correo. Por convención se utiliza un comando LOGIN con el identificador de usuario "anonymous". Se REQUIERE una contraseña. Es dependiente de la implementación, en cuanto a requerimientos, si hay alguno, están en función de la clave y en cuanto a las restricciones de acceso se establecen en los usuarios anónimos.

Una vez autentificado (incluyendo al acceso anónimo), no es posible volver al estado no autentificado.

Además de los comandos universales (CAPABILITY, NOOP, y LOGOUT), los siguientes comandos son considerados válidos en el estado no autentificado: AUTHENTICATE y LOGIN.

6.2.1. Comando AUTHENTICATE

Argumentos: nombre del mecanismo de autentificación

Respuestas: puede solicitarse una continuación de datos

Resultado: OK autentificación completada, ahora estado autentificado
NO fallo de autentificación: mecanismo de autentificación no soportado, credenciales rechazadas
BAD comando desconocido o argumenttos no válidos, transacción de autentificación cancelada.

El comando AUTHENTICATE indica un mecanismo de autentificación, tal y como se describe en [IMAP-AUTH], al servidor. Si el servidor soporta dicho mecanismo de autentificación, lleva a cabo una transacción de protocolo de autentificación e identifica al cliente. Es POSIBLE manejar un mecanismo de protección OPCIONAL para sucesivas interacciones del protocolo. Si no está soportado dicho mecanismo de autentificación, el servidor DEBERIA rechazar el comando AUTHENTICATE enviando una respuesta NO no etiquetada.

El intercambio de protocolo de autentificación consiste de una serie de desafíos de servidor y respuestas del cliente que son específicos del mecanismo de autentificación. Un desafío de servidor consiste en una respuesta de solicitud de continuación de comando con el token "+" seguido de una cadena codificada en BASE64. La respuesta del cliente consiste en una línea constituida de una cadena codificada en BASE64. Si el cliente desea cancelar una transacción de autentificación, elabora una línea con un "*". Si el servidor recibe tal respuesta, DEBE rechazar el comando AUTHENTICATE enviando una respuesta BAD etiquetada.

Un mecanismo de protección ofrece una capa de protección sobre la integridad y privacidad de la conexión. Si se utiliza un mecanismo de protección, este se aplica a todos los datos enviados en sucesivas transmisiones en uso de la conexión actual. Este mecanismo de protección se activa justo a continuación del CRLF que finaliza la transacción de autentificación del cliente, y justo después del CRLF de la respuesta OK etiquetada por parte del servidor. Una vez que el mecanismo de protección está en acción, el flujo de comandos y octetos de respuesta se procesan dentro de buffers de texto cifrado. Cada uno de los buffers se transmiten a lo largo de la conexión como flujos de octetos precedidos de un campo de cuatro octetos en orden de byte de red que representan la longitud del dato siguiente. La longitud máxima del buffer de

texto cifrado viene definido por el mecanismo de protección.

Los mecanismos de autentificación son OPCIONALES. Los mecanismos de protección son también OPCIONALES; un mecanismo de autentificación ES POSIBLE que se implemente sin ningún mecanismo de protección. Si un comando AUTHENTICATE falla con la devolución de una respuesta NO, el cliente ES POSIBLE que intente otro mecanismo de autentificación elaborando otro comando AUTHENTICATE, o ES POSIBLE que intente la autentificación mediante la utilización del comando LOGIN. En otras palabras, el cliente ES POSIBLE que solicite tipos de autentificación en orden de precedencia decreciente, con el comando LOGIN como último recurso.

Ejemplo:
```
S: * OK KerberosV4 IMAP4rev1 Servidor
C: A001 AUTHENTICATE KERBEROS_V4
S: + AmFYig==
C: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
   +nZImJjnTNHJUtxAA+o0KPKfHEcAFs9a3CL5Oebe/ydHJUwYFd
   WwuQ1MWiy6IesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
S: + or//EoAADZI=
C: DiAF5A4gA+oOIALuBkAAmw==
S: A001 OK Kerberos V4 autentificación conseguida.
```

Nota: los espacios en blanco en la primera respuesta del cliente se utilizan en mor de la claridad editorial y no se utilizan en autentificadores reales.

6.2.2. Comando LOGIN

Argumentos: nombre de usuario
contraseña

Respuestas: no hay respuestas especificas para este comando

Resultado:
- OK login completado, ahora en estado autentificado
- NO login fallido: nombre de usuario o contraseña rechazados
- BAD comando desconocido o argumentos no válidos

El comando LOGIN identifica el cliente al servidor y se hace cargo de la clave en texto plano autentificando al usuario.

Ejemplo:
```
C: a001 LOGIN SMITH SESAME
```



```
S: a001 OK LOGIN completado
```

6.3. Comandos de cliente - Estado autentificado

En el estado autentificado, se permite utilizar los comandos que manipulan el buzón de correo como entidades atómicas. De entre estos comandos, los comandos SELECT y EXAMINE seleccionarán un buzón de correo para conseguir acceso y entrada al estado seleccionado.

Además de los comandos universales (CAPABILITY, NOOP, y LOGOUT), los comandos que vemos a continuación son válidos en el estado autentificado: SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, y APPEND.

6.3.1. Comando SELECT

Argumentos: nombre del buzón de correo

Respuestas: respuestas no etiquetadas OBLIGATORIAS: FLAGS, EXISTS, RECENT
respuestas no etiquetadas OPCIONALES: UNSEEN, PERMANENTFLAGS

Resultado: OK selección completada, ahora en estado seleccionado
NO selección fallida, ahora en estado autentificado: no existe el buzón de correo, no es posible acceder al buzón.
BAD comando desconocido o argumentos no válidos

El comando SELECT selecciona un buzón de correo para que en un futuro se pueda tener acceso a los mensajes que este contiene.Antes de enviar un OK al cliente, el servidor DEBE enviar al cliente los datos no etiquetados que se refieren a continuación:

FLAGS banderas especificadas en el buzón de correo. Ver la descripción de las respuestas FLAGS para más información.

<n> EXISTS
Número de mensajes existentes en el buzón de correo. Ver la descripción de la respuesta EXISTS para más información.

<n> RECENT

Número de mensajes con la bandera \Recent activa. Ver descripción de la respuesta RECENT para más información.

OK [UIDVALIDITY <n>]
El valor de vigencia de identificador único. Ver la descripción del comando UID para más información.

para definir el estado inicial del buzón de correo al cliente.

El servidor DEBERIA enviar un código de respuesta UNSEEN dentro de una respuesta no etiquetada OK, indicando el número de secuencia de mensaje del primer mensaje no visto hasta ahora presente en el buzón de correo.

Si el cliente no puede cambiar el estado de una o más de las banderas listadas en la respuesta no etiquetada FLAGS, el servidor DEBERIA enviar un código de respuesta PERMANENTFLAGS dentro de una respuesta no etiquetada OK, listando las banderas que el cliente puede cambiar de forma permanente.

Sólo puede seleccionarse un único buzón de correo por conexión; un acceso simultáneo a múltiples buzones de correo precisa de múltiples conexiones. El comando SELECT deselecciona automáticamente el buzón de correo actual antes de intentar seleccionar uno nuevo. En consecuencia, si se ha seleccionado un buzón de correo y se intenta un comando SELECT fallido, no se selecciona ningún buzón de correo.

Si se permite al cliente modificar el buzón de correo, el servidor DEBERIA preceder al texto de la respuesta etiquetada OK con el código de respuesta "[READ-WRITE]".

Si se permite que el cliente modifique el buzón de correo pero si se le permite un acceso de lectura, el buzón de correo está seleccionado como de solo lectura, y el servidor DEBE preceder el texto de la respuesta etiquetada OK del SELECT con el código de respuesta "[READ-ONLY]". El acceso de solo lectura a través de SELECT difiere del comando EXAMINE en que ciertos buzones de correo de solo lectura ES POSIBLE que permitan cambiar el estado permanente a uno por-usuario (en oposición al global). Los mensajes de noticias de red marcados en un archivo

permanente por-usuario que puede ser modificado con buzones de correo de solo lectura.

Ejemplo:

```
C: A142 SELECT INBOX
S: * 172 EXISTS
S: * 1 RECENTS: * OK [UNSEEN 12]
   El mensaje 12 es el primer mensaje no visto
S: * OK [UIDVALIDITY 3857529045] UIDs válido
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limitado
S: A142 OK [READ-WRITE] SELECT completado
```

6.3.2. Comando EXAMINE

Argumentos: nombre de buzón

Respuestas: respuestas no etiquetadas OBLIGATORIAS: FLAGS,EXISTS, RECENT
respuestas no etiquetadas OK OPCIONALES: UNSEEN, PERMANENTFLAGS

Resultado: OK examine completado, ahora en estado seleccionado
NO examine fallido, ahora en estado autentificado: no existe el buzón de correo, no puede accederse al buzón.
BAD comando desconocido o argumentos no válidos

El comando EXAMINE es idéntico al comando SELECT y devuelve la misma salida; sin embargo, el buzón de correo seleccionado se establece como de solo lectura. No se permiten modificaciones en el estado permanente del buzón de correo, incluyendo el estado por-usuario.

El texto de la respuesta OK etiquetada al comando EXAMINE DEBE comenzar con el código de respuesta [READ-ONLY].

Ejemplo:

```
C: A932 EXAMINE blurdybloop
S: * 17 EXISTS
S: * 2 RECENT
S: * OK [UNSEEN 8] El mensaje 8 es el primero
   que se ha visto
S: * OK [UIDVALIDITY 3857529045] UIDs válido
```

```
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * OK [PERMANENTFLAGS ()] No se permiten banderas
   permanentes
S: A932 OK [READ-ONLY] EXAMINE completado
```

6.3.3. Comando CREATE

Argumentos: nombre de buzón de correo

Respuestas: no hay respuestas especificas para este comando

Resultado: OK creación completada
NO creación fallida: no puede crearse un buzón de correo con este nombre
BAD comando desconocido o argumentos no válidos

El comando CREATE crea un buzón de correo con el nombre especificado. Se devolverá una respuesta OK siempre y cuando haya sido creado este nuevo buzón con el nombre especificado. Cuando intentemos crear un buzón o INBOX con un nombre asociado igual a un nombre de buzón ya existente obtendremos un error. Cualquier error ocurrido durante la creación devolverá una respuesta etiquetada NO.

Si el nombre de buzón de correo contiene como sufijo el carácter separador de la jerarquía del servidor (de la misma manera que cuando se ejecuta un comando LIST), quiere decir que el cliente quiere crear nombres de buzón con el nombre dado, dentro de la jerarquía. Las implementaciones de servidor que no requieren esta declaración DEBEN ignorarla.

Si el carácter separador de jerarquía del servidor aparece en cualquier otro lugar del nombre, el servidor DEBERIA crear los nombres de jerarquía superiores que necesite el comando CREATE para ejecutarse de manera correcta. En otras palabras, un intento de crear "foo/bar/zap" en un servidor cuyo carácter separador de jerarquía es "/" DEBERIA crear foo/ y foo/bar si es que estos no existen ya.

Si se crea un buzón de correo con el mismo nombre que el que poseía un buzón borrado anteriormente, sus identificadores únicos DEBEN ser mayores que cualquiera de los identificadores únicos usados por el buzón de correo anterior a menos que el nuevo buzón posea un valor de vigencia de identificador único diferente. Ver la

descripción del comando UID para más información.

Ejemplo:
```
C: A003 CREATE owatagusiam/
S: A003 OK CREATE completado
C: A004 CREATE owatagusiam/blurdybloop
S: A004 OK CREATE completado
```

Nota: La interpretación de este ejemplo depende de si el carácter separador de jerarquía del servidor es "/". Si "/" lo es, se creará un nuevo nivel en la jerarquía con nombre "watagusiam" que incluirá un miembro llamado

"blurdybloop". Si no lo es, se crearán dos buzones de correo en el mismo nivel de la jerarquía.

6.3.4. Comando DELETE

Argumentos: nombre de buzón de correo

Respuestas: no hay respuestas especificas para este comando

Resultado:
- OK borrado completado
- NO borrado fallido: no puede eliminarse el buzón que posee este nombre.
- BAD comando desconocido o argumentos no válidos

El comando DELETE borra de forma permanente el buzón de correo que posea el nombre especificado. Si este ha sido borrado, se devolverá una respuesta etiquetada NO. Es un error intentar borrar INBOX o cualquier buzón cuyo nombre no exista.

El comando DELETE NO DEBE eliminar nombres de jerarquía inferiores. Por ejemplo, si un buzón llamado "foo" tiene un nombre de jerarquía "foo.bar" inferior (asumiendo que "." es el carácter delimitador de la jerarquía), si se intenta eliminar "foo", "foo.bar" NO DEBE ser eliminado. Cualquier intento de borrar un nombre que posea nombres de jerarquía inferiores y que tenga el atributo de nombre de buzón \Noselect, constituye un error. (ver descripción de la respuesta dada por el comando LIST para más información).

Está permitido eliminar un nombre de jerarquía que posea nombres de jerarquía inferiores pero que carezcan del atributo de nombre de buzón \Noselect. En este caso, se borrarán todos los mensajes del buzón, y este nombre

recibirá el atributo de nombre de buzón \Noselect.

El valor más alto de identificador único del buzón borrado con anterioridad DEBE preservarse para que si alguna vez se crea un buzón con el mismo nombre este no utilice los identificadores del buzón anterior, a menos que el nuevo buzón posea un valor de vigencia de identificador único diferente. Ver la descripción del comando UID para más información.

Ejemplos:

```
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/" foo
S: * LIST () "/" foo/bar
S: A682 OK LIST completado
C: A683 DELETE blurdybloop
S: A683 OK DELETE completado
C: A684 DELETE foo
S: A684 NO nombre "foo" tiene nombres de
jerarquía inferiores.
C: A685 DELETE foo/bar
S: A685 OK DELETE completado
C: A686 LIST "" *
S: * LIST (\Noselect) "/" foo
S: A686 OK LIST completado
C: A687 DELETE foo
S: A687 OK DELETE completado
```

```
C: A82 LIST "" *
S: * LIST () "." blurdybloop
S: * LIST () "." foo
S: * LIST () "." foo.bar
S: A82 OK LIST completado
C: A83 DELETE blurdybloop
S: A83 OK DELETE completado
C: A84 DELETE foo
S: A84 OK DELETE completado
C: A85 LIST "" *
S: * LIST () "." foo.bar
S: A85 OK LIST completado
C: A86 LIST "" %
S: * LIST (\Noselect) "." foo
S: A86 OK LIST completado
```

6.3.5. Comando RENAME

Argumentos: nombre del buzón existente
nombre del nuevo buzón

Respuestas: no hay respuestas específicas para este comando

Resultado: OK cambio de nombre completado
NO cambio de nombre: no puede cambiarse el nombre de buzón por este nombre.
BAD comando desconocido o argumentos no válidos

El comando RENAME cambia el nombre del buzón de correo. Se devolverá una respuesta etiquetada OK sólo si el nombre del buzón de correo ha sido modificado. Es un error intentar renombrar un buzón desde un buzón cuyo nombre no exista o utilizar un nombre de buzón correo que ya exista. Cualquier error relacionado con el cambio de nombres de buzón de correo devolverá una respuesta etiquetada NO.

Si dicho nombre posee nombres de jerarquía inferiores, estos también DEBEN cambiarse de nombre. Por ejemplo, si cambiamos el nombre "foo" por "zap" provocará que "foo/bar" (asumimos que "/" es el carácter separador de jerarquía) se convierta en "zap/bar".Se DEBE preservar el identificador único más alto que se haya utilizado en el buzón anterior, para que en el caso de que se cree un nuevo buzón con el mismo nombre que el anterior evitar la utilización de los identificadores propios del buzón anterior, a menos que el buzón anterior posea un valor de vigencia de identificador único diferente. Ver la descripción del comando UID para más información.

Se permite el cambio de nombres de INBOX, y posee un comportamiento especial: se crea un nuevo buzón de correo con el nombre especificado que contiene todos los mensajes presentes en INBOX y los elimina de INBOX. Si la implementación del servidor permite nombres de jerarquía inferiores para INBOX, estos no se ven afectados por un cambio de nombre del INBOX.

Ejemplo:

```
C: A682 LIST "" *
S: * LIST () "/" blurdybloop
S: * LIST (\Noselect) "/"foo
S: * LIST () / foo/bar
S: A682 OK LIST completado
C: A683 RENAME blurdybloop sarasoop
S: A683 OK RENAME completado
```

```
C: A684 RENAME foo zowie
S: A684 OK RENAME completado
C: A685 LIST  "" *
```



```
S: * LIST () "/" sarasoop
S: * LIST (\Noselect) "/" zowie
S: * LIST () "/" zowie/bar
S: A685 OK LIST completado
C: Z432 LIST  "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: Z432 OK LIST completado
C: Z433 RENAME INBOX old-mail
S: Z433 OK RENAME completado
C: Z434 LIST "" *
S: * LIST () "." INBOX
S: * LIST () "." INBOX.bar
S: * LIST () "." old-mail

S: Z434 OK LIST completado
```

6.3.6. Comando SUBSCRIBE

Argumentos: buzón de correo

Respuestas: no hay respuestas especificas para este comando

Resultado:
- OK comando completado
- NO suscripción fallida: no puede asignarse dicho nombre
- BAD comando desconocido o argumentos no válidos

El comando SUBSCRIBE añade el nombre de buzón de correo especificado al conjunto de buzones "activos" o "suscritos" presentes en el servidor, conjunto que se lista tras la ejecución del comando LSUB. Este comando devuelve una respuesta OK etiquetada únicamente si la suscripción tiene éxito.

Un servidor ES POSIBLE que valide el argumento del buzón de correo utilizado en SUBSCRIBE para verificar que este existe. Sin embargo, no DEBE eliminarse de forma unilateral un nombre de buzón de correo existente de la lista de suscripción incluso si el buzón de correo que posee ese nombre no existe.

> Nota: esto es necesario puesto que algunos sitios de servidor mantienen tareas rutinarias que eliminan buzones con nombres bien conocidos (p.ej. "system-alerts") tras la

> pérdida de interés de su contenido, con la intención de volver a crearlo con contenido actualizado o apropiado al momento actual.

Ejemplo:

```
C: A002 SUBSCRIBE #news.comp.mail.mime
S: A002 OK SUBSCRIBE completado
```

6.3.7. Comando UNSUBSCRIBE

Argumentos: nombre de buzón de correo

Respuestas: no hay respuestas especificas para este comando

Resultado:
- OK abandonar suscripción completado
- NO abandonar suscripción fallido: no puede darse de baja ese nombre
- BAD comando desconocido o argumentos no válidos

El comando UNSUBSCRIBE elimina el nombre de buzón de correo especificado del conjunto de buzones "activos" o "suscritos" del servidor que se muestran con el comando LSUB. Este comando devuelve una respuesta OK etiquetada únicamente cuando el comando tiene éxito.

Ejemplo:
```
C: A002 UNSUBSCRIBE #news.comp.mail.mime
S: A002 OK UNSUBSCRIBE completado
```

6.3.8. Comando LIST

Argumentos: nombre de referencia
nombre del buzón con los posibles comodines

Respuestas: respuestas no etiquetadas: LIST

Resultado:
- OK listado completado
- NO- listado fallido: no puede listarse el nombre oreferencia
- BAD comando desconocido o argumentos no válidos

El comando LIST devuelve un subconjunto de nombres del conjunto global de todos los nombres disponibles, al cliente. Se devuelven cero o más respuestas, conteniendo los atributos del nombre, delimitador de jerarquía, y nombre; ver la descripción de la respuesta LIST para más información.

El comando LIST DEBERIA devolver sus datos con rapidez, sin retraso. Por ejemplo, NO DEBERIA tener demasiados problemas en calcular el estado \Marked o \Unmarked o llevar a cabo cualquier otro procesamiento; si cada nombre requiere 1 segundo de procesamiento, entonces un listado de 1200 nombres supondría !20 minutos!

Un argumento de nombre de referencia vacío (cadena "") indica que el nombre del buzón se interpreta como en un SELECT. Los nombres de buzón devueltos DEBEN coincidir con el patrón de nombre especificado. Un argumento de nombre de referencia no vacío es el nombre de un buzón o un nivel de la jerarquía del buzón, e indica un contexto en el cual la interpretación del nombre del buzón se realiza en base a la implementación.

Un argumento de nombre de buzón vacío (cadena "") supone una solicitud especial que consiste en la devolución del delimitador de jerarquía y el nombre del root asociado al nombre dado en la referencia. El valor devuelto como root ES POSIBLE que sea null si la referencia no es en función del root o es null. En cualquier caso, se devolverá el delimitador de jerarquía. Esto permite que un cliente

obtenga el delimitador de jerarquía incluso cuando no existan buzones que posean el nombre especificado.

Los argumentos referencia y nombre de buzón son interpretados en función de la implementación, en forma canónica que representa una jerarquía de izquierda a derecha no ambigua. Los nombre de buzón devueltos estarán en forma interpretada.

Cualquier parte del argumento referencia incluida en la forma interpretada DEBERIA preceder a la misma. DEBERIA encontrarse en el mismo formato que el argumento nombre de referencia. Esta norma permite que el cliente conozca si el nombre de buzón devuelto se encuentra en el contexto del argumento de referencia, o si el argumento de buzón anuló el argumento de referencia. Sin ayuda de esta norma, el cliente tendría que conocer la semántica de nombres utilizada en el servidor incluyendo cuales son los caracteres "ruptura" que anulan un contexto de nombres.

Por ejemplo, aquí tenemos algunos ejemplos de cómo pueden interpretarse en un servidor basado en UNIX, las referencias y nombre de buzón:

Referencia	Nombre de buzón	Interpretación
~smith/Mail/	foo.*	~smith/Mail/foo.*
archive/	%	archive/%
#news.	comp.mail.*	#news.comp.mail.*
~smith/Mail/	/usr/doc/foo	/usr/doc/foo
archive/	~fred/Mail/*	~fred/Mail/*

Los primeros tres ejemplos muestran cómo se produce la interpretación en el contexto del argumento referencia. Ha de observarse que "~smith/Mail" NO DEBERIA ser transformado en algo como "/u2/users/smmith/Mail", o resultaría imposible para el cliente determinar si la interpretación se hizo en el contexto de la referencia.

El carácter "*" es un comodín, y equivale a un patrón que aceptacero o más caracteres en dicha posición. El carácter "%" es similar a "*", pero este no acepta un carácter delimitador de jerarquía. Si el carácter comodín "%" es el último carácter de un argumento nombre de buzón, se devolverán también todas los niveles de jerarquía coincidentes. Si estos niveles son buzones no seleccionables, se devolverán con el atributo de nombre de buzón \Noselect (ver la descripción de la respuesta LIST para más información).

Las implementaciones de servidor permiten "esconder" buzones accesibles de la inclusión de caracteres comodín, evitando de esta forma que determinados nombres o caracteres coincidan con un carácter comodín en ciertas situaciones. Por ejemplo, un servidor basado en UNIX puede restringir la forma de interpretar el carácter "*" para que un carácter "/" inicial no coincida.

El nombre especial INBOX se incluye en la salida de LIST, si el servidor soporta INBOX para este usuario en concreto y si la cadena "INBOX" con mayúsculas concuerda con los argumentos referencia y nombre de buzón ya interpretados con los caracteres comodín en la forma descrita más abajo. El criterio para omitir INBOX es si un SELECT INBOX devolverá un error; no es relevante si el INBOX del

usuario real reside en este u otro servidor.

```
Ejemplo:        C: A101 LIST "" ""
                S: * LIST (\Noselect) "/" ""
              S: A101 OK LIST completado
```



```
                   C: A102 LIST #news.comp.mail.misc ""
                 S: * LIST (\Noselect) "." #news.
                   S: A102 OK LIST completada
                   C: A103 LIST /usr/staff/jones ""
                   S: * LIST (\Noselect) "/" /
                   S: A103 OK LIST completado
                   C: A202 LIST ~/Mail/ %
                   S: * LIST (\Noselect) "/"~/Mail/foo
                   S: * LIST () "/"~/Mail/meetings
                   S: A202 OK LIST completado
```

6.3.9. Comando LSUB

```
   Argumentos:     nombre de referencia
                   nombre de buzón con posibles comodines

   Respuestas:     OK  lsub completado
                   NO  lsub fallido: no puede listarse la
                        referencia o el nombre
                   BAD  comando desconocido o argumentos no
                        válidos
```

El comando LSUB devuelve un subconjunto de nombres del conjunto de nombres declarados por el usuario como "activos" o "suscritos". Se devolverán cero o más respuestas LSUB no etiquetadas. Los argumentos de LSUB tienen el mismo formato que en el comando LIST.

Un servidor ES POSIBLE que valide los nombres suscritos para comprobar que estos aun existen. Si uno de los nombres no existe, esto se DEBERIA indicar incluyendo el atributo \Noselect en la respuesta LSUB.

El servidor NO DEBE eliminar de forma unilateral un nombre de buzón existente de la lista de suscripción incluso cuando el buzón que posea dicho nombre ya no exista.

```
   Ejemplo:        C: A002 LSUB "#news." "comp.mail.*"
                   S: * LSUB () "." #news.comp.mail.mime
                   S: * LSUB () "." #news.comp.mail.misc
                   S: A002 OK LSUB completado
```

6.3.10. Comando STATUS

```
   Argumentos:     nombre de buzón
                   nombres de elemento de datos de estado

   Respuestas:     respuestas no etiquetadas: STATUS
```



```
   Resultado:      OK  comprobación de estado completado
```

NO comprobación de estado fallido: el nombre no posee estado
BAD comando desconocido o argumentos no válidos

El comando STATUS solicita el estado del buzón especificado. No produce ningún cambio en el buzón seleccionado, ni afecta al estado de los mensajes contenidos en el buzón (en particular, STATUS NO DEBE provocar la pérdida de la bandera \Recent).

El comando STATUS ofrece una solución alternativa para la apertura de una segunda conexión IMAP4rev1 y generar un comando EXAMINE sobre el buzón para averiguar su estado sin tener que deseleccionar el buzón actual de la primera conexión IMAP4rev1.

A diferencia del comando LIST, el comando STATUS no garantiza que las respuestas que este produzca se ejecuten con rapidez. En algunas implementaciones, el servidor está obligado a abrir el buzón en modo solo lectura internamente para obtener cierta información de estado. También al contrario que el comando LIST, el comando STATUS no acepta comodines.

Los elementos de datos de estado que están definidos actualmente y pueden ser solicitados son:

MESSAGES El número de mensajes que contiene el buzón.

RECENT El número de mensajes con la bandera \Recent activa.

UIDNEXT El valor de UID que le será asignado al primer mensaje nuevo que llegue al buzón. Se garantiza que este valor no cambiará a menos que se añadan nuevos mensajes al buzón; si esto sucede entonces este valor será modificado incluso si estos mensajes nuevos son expulsados o borrados posteriormente.

UIDVALIDITY Valor de vigencia para el identificador único del buzón

UNSEEN El número de mensajes que no tienen la bandera \Seen activada.

Ejemplo:

```
C: A042 STATUS blurdybloop (UIDNEXT MESSAGES)
S: * STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
S: A042 OK STATUS completado
```

6.3.11. Comando APPEND

Argumentos: nombre de buzón
lista entre paréntesis de banderas OPCIONAL
cadena fecha/hora OPCIONAL
literal mensaje

Respuestas: no hay respuestas especificas para el comando

Resultado: OK acción de añadido completada
NO acción de añadido error: no puede añadirse al buzón, error en banderas o en fecha/hora o en texto del mensaje
BAD comando desconocido o argumentos no válidos

El comando APPEND añade el argumento literal como un nuevo mensaje al final del buzón destino especificado. Este argumento DEBERIA estar en formato de mensaje [RFC-822]. Se permiten caracteres de 8-bit en el mensaje. La implementación de servidor que sea incapaz de preservar de forma adecuada los datos de 8-bit DEBE ser capaz de convertir los datos APPEND de 8-bit a 7-bit usando una codificación de transferencia de contenidos [MIME-IMB].

Nota: ES POSIBLE que haya excepciones, p.ej. borradores de mensaje, en los cuales las líneas de cabecera necesarias se omiten en el argumento literal de mensaje de APPEND. Las razones para hacer esto DEBEN ser sopesadas cuidadosamente.

Si se especifica una lista entre paréntesis de banderas, las banderas DEBERIAN establecerse en el mensaje resultante; de otra manera, la lista de banderas del mensaje resultante estaría vacía por defecto.

Si se especifica una fecha_hora, la fecha interna DEBERIA

establecerse en el mensaje resultante; de otra forma, la fecha interna del mensaje resultante ser la fecha y hora actuales por defecto.

Si el comando append no es eficaz por cualquier razón, el estado del buzón previo a la ejecución del comando APPEND debe ser restaurado; no se permiten añadidos parciales.

Si el buzón destino no existe, un servidor DEBE devolver un error, y NO DEBE crear el buzón de forma automática. A menos que no puedacrearse el buzón destino, el servidor DEBE enviar el código de respuesta "[TRYCREATE]" como prefijo del texto de la respuesta etiquetada NO. Esto indica al cliente que puede intentar ejecutar un comando CREATE y volver a ejecutar el APPEND si el comando CREATE anterior tiene éxito.

Si el buzón está seleccionado, DEBERIAN tener lugar las acciones asociadas a la llegada de nuevos mensajes. Más específicamente, el servidor DEBERIA notificarlo al cliente de manera inmediata mediante una respuesta EXISTS no etiquetada. Si el servidor no lo hace, el cliente puede ejecutar un comando NOOP (o si este falla, un comando CHECK) tras uno o más comandos APPEND.

Ejemplo:

```
C: A003 APPEND mensajes almacenados (\Seen) {310}
C: Fecha: Mon, 7 Feb 1994 21:52:25 0800 (PST)
C: Desde: Fred Foobar <foobar@Blurdybloop.COM>
C: Asunto: reunión por la tarde
C: Destinado a: mooch@owatagu.siam.edu
C: Identificador-mensaje: <B27397-0100000@Blurdybloop.COM>
C: Versión MIME: 1.0
C: Tipo de contenido: TEXT/PLAIN; CHARSET=US-ASCII
C:
```

```
          C: Hola Joe, podemos quedar mañana a las 3:30?
          C:
          S: A003 OK APPEND completado
```

Nota: el comando APPEND no se utiliza para el envío de mensajes, porque no ofrece un mecanismo de transferencia de información de envoltura [SMTP].

6.4. Comandos de cliente - Estado seleccionado

En el estado seleccionado, están permitidos los comandos que manipulan mensajes en un buzón.Además de los comandos universales (CAPABILITY, NOOP, y LOGOUT), y los comandos de

estado autentificado (SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, y APPEND), los comandos siguientes son válidos en el estado seleccionado: CHECK, CLOSE, EXPUNGE, SEARCH, FETCH, STORE, COPY, y UID.

6.4.1. Comando CHECK

Argumentos: ninguno

Respuestas: no hay respuestas especificas para este comando

Resultado: OK comprobación completada
BAD comando desconocido o argumentos no válidos

El comando CHECK solicita un punto de comprobación en el buzón seleccionado. Un punto de comprobación hace referencia a un proceso de mantenimiento dependiente de la implementación asociado al buzón (p.ej. compara el estado en memoria del buzón en el servidor con el estado en disco) que normalmente no se ejecuta por comando. El establecimiento de un punto de comprobación ES POSIBLE que suponga una cantidad de tiempo no despreciable. Si una implementación de servidor no contempla estas consideraciones de mantenimiento, el comando CHECK es equivalente a NOOP.

No está garantizado que como resultado de la ejecución de un comando CHECK obtengamos una respuesta no etiquetada EXISTS. El comando NOOP DEBERIA ser utilizado para el sondeo de nuevo correo, en lugar de CHECK.

Ejemplo: C: FXXZ CHECK
S: FXXZ OK CHECK completado

6.4.2. Comando CLOSE

Argumentos: ninguno

Respuestas: no hay respuestas especificas para este comando

Resultado: OK cierre completado, ahora en estado autentificado
NO cierre fallido: ningún buzón seleccionado
BAD comando desconocido o argumentos no válidos

El comando CLOSE elimina de manera permanente todos los mensajes del buzón que tengan la bandera \Deleted activada, y cambia el estado del buzón de seleccionado a autentificado. No se envía ninguna respuesta EXPUNGE no etiquetada.

Si se selecciona el buzón mediante un comando EXAMINE o en modo solo lectura no se eliminará ningún mensaje, y no se producirá ningún error.

Incluso si el buzón está seleccionado, ES POSIBLE ejecutar los comandos SELECT, EXAMINE o LOGOUT sin haber ejecutado previamente un CLOSE. Los comandos SELECT, EXAMINE y LOGOUT cierran implícitamente el buzón seleccionado sin necesidad de ejecutar un borrado (expunge). Sin embargo, cuando se borran muchos mensajes, una secuencia CLOSE-LOGOUT o CLOSE-SELECT es considerablemente más rápida que una secuencia EXPUNGE-LOGOUT o EXPUNGE-SELECT puesto que no se envían respuestas EXPUNGE no etiquetadas (las cuales el cliente ignoraría).

Ejemplo: C: A341 CLOSE
S: A341 OK CLOSE completado

6.4.3. Comando EXPUNGE

Argumentos: ninguno

Respuestas: respuestas no etiquetadas: EXPUNGE

Resultado: OK borrado completado
NO borrado fallido: no puede borrarse (p.ej. permiso denegado)
BAD comando desconocido o argumentos no válidos

El comando EXPUNGE borra permanentemente todos los mensajes del buzón que tengan la bandera \Deleted activada. Antes de devolver un OK al cliente, se envía una respuesta no etiquetada EXPUNGE por cada mensaje que haya sido eliminado.

Ejemplo: C: A202 EXPUNGE
S: * 3 EXPUNGE
S: * 3 EXPUNGE
S: * 5 EXPUNGE
S: * 8 EXPUNGE
S: A202 OK EXPUNGE completado

Nota: en este ejemplo, los mensajes 3,4,7, y 11 tenían la bandera \Deleted activada. Ver descripción de la respuesta EXPUNGE para más información.

6.4.4. Comando SEARCH

Argumentos: especificación [CHARSET] OPCIONAL
patrón de búsqueda (uno o más)

Respuestas: respuesta no etiquetada REQUERIDA

Resultado: OK búsqueda completada
NO error en búsqueda: no puedo buscar por

ese [CHARSET] o por el patrón
BAD comando desconocido o argumentos no válidos

El comando SEARCH busca los mensajes del buzón que coincidan con el patrón de búsqueda especificado. Este consiste en una o más claves de búsqueda. La respuesta no etiquetada SEARCH del servidor contiene un listado de números de secuencia de mensaje que se corresponden con aquellos mensajes que coinciden con el patrón de búsqueda.

Cuando se especifican múltiples claves, el resultado es la intersección (función AND) de todos los mensajes que coinciden con dichas claves. Por ejemplo, el patrón DELETED FROM "SMITH"SINCE 1-Feb-1994 se refiere a todos los mensajes enviados por Smith que han sido eliminados y fueron recibidos en el buzón desde el 1 de Febrero de 1994. Una clave de búsqueda puede consistir también en una lista entre paréntesis que contenga una o más claves de búsqueda. (p.ej. para el uso con las claves OR o NOT).

Las implementaciones de servidor ES POSIBLE que excluyan fragmentos del cuerpo con tipos multimedia distintos de TEXT o MESSAGE del patrón de búsqueda.

La especificación OPCIONAL de [CHARSET] está constituida por la palabra "CHARSET" seguida de una [tabla de caracteres] registrada. Esto indica la [tabla de caracteres] utilizada en las cadenas que aparecen en el patrón de búsqueda. Las codificaciones de transferencia de contenido [MIME-IMB], y cadenas [MIME-HDRS] en las cabeceras [RFC-822]/[MIME-IMB], DEBEN ser decodificadas antes de comparar texto con una [tabla de caracteres] distinta del US-ASCII. Se DEBE tener soporte para el US-ASCII; también ES POSIBLE que se permitan otras [tabla

de caracteres] . Si el servidor no soporta la [tabla de caracteres] especificada, DEBE devolver una respuesta etiquetada NO (no una respuesta BAD).

En todas las claves de búsqueda que contengan cadenas, un mensaje coincide con la clave si la cadena es una subcadena del campo.

El patrón no hace distinción entre mayúsculas-minúsculas.

Las claves de búsqueda definidas son como sigue. Ver la sección Sintaxis Formal para una definición sintáctica más precisa de los argumentos.

<conjunto de mensajes> Mensajes con números de secuencia de mensaje que se corresponden al conjunto de números de secuencia de mensaje especificado

ALL Todos los mensajes del buzón; la clave inicial por defecto para la intersección (AND).

ANSWERED Mensajes con la bandera \Answered activada.

BCC <cadena> Mensajes que contienen la cadena especificada en el campo BCC de la estructura del envoltorio.

BEFORE <fecha> Mensajes cuya fecha interna es anterior a la fecha especificada.

BODY <cadena> Mensajes que contienen la cadena especificada en el cuerpo del mensaje.

CC <cadena> Mensajes que contienen la cadena especificada en el campo CC de la estructura de envoltorio.

DELETED Mensajes con la bandera \Deleted activada.

DRAFT Mensajes con la bandera \Draft activada

FLAGGED Mensajes con la bandera \Flagged

activada.

FROM <cadena> Mensajes que contienen la cadena especificada en el campo FROM de la estructura de envoltorio.

HEADER <nombre del campo><cadena>
Mensajes cuya cabecera posee el nombre de campo especificado (como se define en [RFC-822]) y que contiene la cadena especificada en el campo cuerpo.

KEYWORD <bandera> Mensajes con el conjunto de claves especificado

LARGER <n> Mensajes con un tamaño [RFC-822] mayor que el número de octetos especificado.

NEW Mensajes cuya bandera \Recent está activada pero no lo está la bandera \Seen. Esto es equivalente en funcionalidad a "(RECENT UNSEEN)".

NOT <clave de búsqueda>
Mensajes que no coinciden con la clave de búsqueda especificada.

OLD Mensajes cuya bandera \Recent no está activada. Esto es equivalente en funcionalidad a "NOT RECENT" (lo contrario de "NOT NEW").

ON <fecha> Mensajes cuya fecha interna está dentro de la fecha especificada.

OR <clave de búsqueda1><clave de búsqueda2>
Mensajes que coinciden con ambas claves

RECENT Mensajes cuya bandera \Recent está activada.

SEEN Mensajes cuya bandera \Seen está activada.

SENTBEFORE <fecha> Mensajes cuya Fecha [RFC-822]:

cabecera anterior a la fecha especificada.

SENTON <fecha> Mensajes cuya Fecha [RFC-822]: cabecera está dentro de la cabecera especificada.

SENTSINCE <fecha> Mensajes cuya Fecha [RFC-822]: cabecera está dentro o es posterior a la fecha especificada.

SINCE <fecha> Mensajes cuya fecha interna está dentro o es posterior a la fecha especificada.

SMALLER <fecha> Mensajes de un tamaño [RFC-822] menor queel número de octetos especificado.

SUBJECT <cadena> Mensajes que contienen la cadena especificada en el campo SUBJECT de la estructura de envoltorio.

TEXT <cadena> Mensajes que contienen la cadena especificada en la cabecera o cuerpo del mismo.

TO <cadena> Mensajes que contienen la cadena especificada en el campo TO de la estructura de envoltorio.

UID <conjunto de mensajes> Mensajes con identificadores únicos quese corresponden con el conjunto de identificadores únicos especificado.

UNANSWERED Mensajes cuya bandera \Answered no está activada.

UNDELETED Mensajes cuya bandera \Deleted no está activada.

UNDRAFT Mensajes cuya bandera \Draft no está activada.

UNFLAGGED Mensajes cuya bandera \Flagged no está activada.

UNKEYWORD <bandera> Mensajes que no poseen el conjunto de claves especificado.

UNSEEN Mensajes cuya bandera \Seen no está activada.

Ejemplo:

```
        C: A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM Smith
        S: * SEARCH 2 84 882
        S: A282 OK SEARCH completado
```

6.4.5. Comando FETCH

```
   Argumentos:     conjunto de mensajes
                   nombres de elementos de datos de mensaje

   Respuestas:     respuestas no etiquetadas: FETCH

   Resultado:      OK  recuperación completada
                   NO  error en recuperación: no puede
                        encontrarse el dato
                   BAD comando desconocido o argumentos no
                        válidos
```

El comando FETCH recupera datos asociados a un mensaje contenido en el buzón. Los elementos de datos que pueden obtenerse son un único átomo o una lista entre paréntesis.

Los elementos de datos definidos actualmente que pueden recuperarse mediante FETCH son:

ALL
: Macro equivalente a: (FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)

BODY
: Forma no extensible de BODYSTRUCTURE

BODY[<sección>]<<parcial>>
: El texto de una sección del cuerpo en particular. La forma de especificar la sección es un conjunto de cero o más especificadores de parte delimitados por puntos. Un especificador de parte es o un número de parte o uno de los siguientes especificadores: HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, MIME, y TEXT. Una especificación de sección vacía se refiere al mensaje al completo, incluyendo la cabecera.

Todos los mensajes tienen al menos un número de parte.Los mensajes no [MIME-IMB], y los mensajes no multiparte [MIME-IMB] sin mensaje encapsulado, solo tienen una parte 1.

A los mensajes multiparte se les asignan números de parte consecutivos, de la misma forma en que estas se sitúan en el propio mensaje. Si una parte en particular es de tipo mensaje o multiparte, su parte DEBE ser indicada por un punto seguido de un número de parte dentro de la parte anidada en la multiparte.

Una parte de tipo MESSAGE/RFC822 también tiene números de parte anidados, que se refieren a partes del cuerpo de parte del MESSAGE.

Los especificadores de parte HEADER, HEADER.FIELDS, HEADER.FIELDS.NOT, y TEXT pueden formar por sí solos un especificador de parte o pueden ir precedidos de uno o más especificadores de parte numéricos, teniendo en cuenta que los especificadores de parte numéricos se refieren a una parte del tipo MESSAGE/RFC822. El especificador de parte MIME DEBE ir precedido de uno o más especificadores de parte numéricos.

Los especificadores de parte HEADER, HEADER.FIELDS, y HEADER.FIELDS.NOT se refieren a la cabecera [RFC-822] del mensaje o de un mensaje MESSAGE/RFC822 encapsulado [MIME-IMT]. Tras HEADER.FIELDS y HEADER.FIELD.NOT se sitúa una lista de nombres nombre-campo (definido en [RFC-822]), y devuelve un subconjunto de la cabecera. El subconjunto devuelto por HEADER.FIELDS contiene solo aquellos campos de cabecera con un camponombre que coincida con alguno de los incluidos en la lista; de igual manera, el subconjunto devuelto por HEADER.FIELDS.NOT contiene solo los campos de cabecera que no coincidan con ningún campo-nombre de la lista. No hay distinción entre mayúsculas-minúsculas. En cualquier caso, siempre se incluye una línea en blanco entre la cabecera y el cuerpo.

El especificador de parte MIME se refiere a la cabecera de esta parte [MIME-IMB].

El especificador de parte TEXT se refiere al cuerpo del texto del mensaje, omitiendo la cabecera

[RFC-822].

Aquí, tenemos un ejemplo de un mensaje complejo con alguno de sus especificadores de parte:

```
 HEADER       ([RFC-822] cabecera del mensaje)
 TEXT         MULTIPART/MIXED
 1            TEXT/PLAIN
 2            APPLICATION/OCTET-STREAM
 3            MESSAGE/RFC822
3.HEADER     ([RFC-822] cabecera del mensaje)
 3.TEXT       ([RFC-822] cuerpo del texto del
              mensaje)
 3.1          TEXT/PLAIN
 3.2          APPLICATION/OCTET-STREAM
 4            MULTIPART/MIXED
 4.1          IMAGE/GIF
 4.1.MIME     ([MIME-IMB] cabecera para la
              imagen/GIF)
 4.2          MESSAGE/RFC822
 4.2.HEADER   ([RFC-822] cabecera del mensaje)
4.2.TEXT     ([RFC-822] cuerpo del texto del
              mensaje)
4.2.1.       TEXT/PLAIN
 4.2.2        MULTIPART/ALTERNATIVE
 4.2.2.1      TEXT/PLAIN
 4.2.2.2      TEXT/RICHTEXT
```

Es posible recuperar una subcadena del texto designado. Esto se hace añadiendo un ("<"), la posición del octeto del primer octeto que deseamos que sea el primero, y un (">") en el especificador de parte. Si el primer octeto está más allá del final del texto, se devuelve una cadena vacía.

Cualquier intento de recuperación parcial más allá del final del texto es truncado para que se ejecute de manera correcta. Una búsqueda que comience en el octeto 0 se toma como una búsqueda parcial, incluso si esta ya ha sido truncada.

Nota: esto significa que BODY[]<0.2048> de un

mensaje de 1500 octetos devolverá BODY[]<0>
con un literal de tamaño 1500, no BODY[].

Nota: una recuperación de subcadena de un
especificador de parte HEADER.FIELDS o
HEADER.FIELDS.NOT se calcula tras establecer

la cabecera.

Implícitamente la bandera \Seen está activada; si
esto provoca que el estado de las banderas cambie
DEBERIAN incluirse como parte de las respuestas
FETCH.

BODY.PEEK[<sección>]<<parcial>>
Una forma alternativa de BODY[<sección>] que no
establece implícitamente la bandera \Seen como
activa.

BODYSTRUCTURE
La estructura [MIME-IMB] del cuerpo del mensaje.
Esto es procesado por el servidor analizando los
campos cabecera de la cabecera [RFC-822] y
cabeceras [MIME-OMB].

ENVELOPE
La estructura del envoltorio del mensaje. Esto se
procesa por el servidor analizando la cabecera
[RFC-822] en las partes de componente, dejando por
defecto tantos campos como sea necesario.

FAST Macro equivalente a: (FLAGS INTERNALDATE
RFC822.SIZE)

FLAGS Las banderas que están activadas para este mensaje.

FULL Macro equivalente a: (FLAGS INTERNALDATE
RFC822.SIZE ENVELOPE BODY)

INTERNALDATE
La fecha interna del mensaje.

RFC822 Funcionalidad equivalente a BODY[], difieren en la
sintaxis de los datos FETCH no etiquetados
resultantes (se devuelve RFC822).

RFC822.HEADER
Funcionalidad equivalente a BODY.PEEK[HEADER],
difieren en la sintaxis de los datos FETCH no
etiquetados resultantes (se devuelve
RFC822.HEADER).

RFC822.SIZE

El tamaño del mensaje [RFC-822].

RFC822.TEXT

Funcionalidad equivalente a BODY[TEXT], difieren en la sintaxis de los datos FETCH no etiquetados resultantes (se devuelve RFC822.TEXT).

UID Identificador único del mensaje.

Ejemplo:

```
C: A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
S: * 2 FETCH ....
S: * 3 FETCH ....
S: * 4 FETCH ....
S: A654 OK FETCH completado
```

6.4.6. Comando STORE

```
Argumentos:     conjunto de mensajes
                nombre de elemento de datos del mensaje
                valor de elemento de datos del mensaje

Respuestas:     respuestas no etiquetadas: FETCH

Resultado:      OK  almacenamiento completado
                NO  error en almacenamiento: no puedo
                     almacenar ese dato
                BAD  comando desconocido o argumentos no
                     válidos
```

El comando STORE modifica los datos asociados con un mensaje contenido en un buzón. Normalmente, STORE devolverá el valor actualizado de los datos con una respuesta FETCH no etiquetada. El sufijo ".SILENT" en el nombre de elemento de datos evita la respuesta FETCH no etiquetada, y el servidor DEBERIA asumir que el cliente ha determinado el valor actualizado por sí mismo o que no le preocupa cual es este valor actualizado.

Nota: dejando a un lado la utilización o no del sufijo ".SILENT", el servidor DEBERIA enviar una respuesta no etiquetada FETCH si detecta un cambio en las banderas del mensaje realizado por un agente externo. Su propósito es que el estado de las banderas sea

establecido sin una condición de prueba.

Los elementos de datos que pueden ser almacenados actualmente son:

FLAGS <lista de banderas>
Sustituye las banderas del mensaje por el argumento. Se devolverá el nuevo valor de las banderas como si realmente se hubiera realizado un FETCH sobre las mismas.

FLAGS.SILENT <lista de banderas>
Equivalente a FLAGS, pero sin devolver un nuevo valor.

+FLAGS <lista de banderas>
Añadir el argumento a las banderas del mensaje. Se devolverá el nuevo valor de las banderas como si realmente se hubiera realizado un FETCH sobre las mismas.

+FLAGS.SILENT <lista de banderas>
Equivalente a +FLAGS, pero sin devolver un nuevo valor.

-FLAGS <lista de banderas>
Elimina el argumento de las banderas del mensaje. Se devolverá el nuevo valor de las banderas como si realmente se hubiera realizado un FETCH sobre las mismas.

-FLAGS.SILENT <lista de banderas>
Equivalente a -FLAGS, pero sin devolver unnuevo valor.

Ejemplo:

```
C: A003 STORE 2:4 +FLAGS (\Deleted)
S: * 2 FETCH FLAGS (\Deleted \Seen)
S: * 3 FETCH FLAGS (\Deleted)
S: * 4 FETCH FLAGS (\Deleted \Flagged \Seen)
S: A003 OK STORE completado
```

6.4.7. Comando COPY

Argumentos: conjunto de mensajes
nombre del buzón

Respuestas: no hay respuestas especificas para este comando

Resultado:
OK copia completado
NO error en copia: no puedo copiar los mensajes o al destino especificado
BAD comando desconocido o argumentos no válidos

El comando COPY copia el/los mensaje/s especificado/s al final del buzón indicado. Las banderas y fecha interna de el/los mensaje/s DEBERIAN ser conservados en la copia.

Si el buzón destino no existe, el servidor DEBERIA devolver un error. No se DEBERIA crear el buzón automáticamente. A menos que sepamos con seguridad que no puede crearse un nuevo buzón, el servidor DEBE enviar un código de respuesta "[TRYCREATE]" como prefijo del texto de la respuesta etiquetada NO. Esto hace saber al cliente que puede intentar ejecutar un comando CREATE y volver a ejecutar el comando COPY si el comando anterior tuvo éxito.

Si el comando CREATE falla por cualquier razón, las implementaciones de servidor DEBEN restaurar el estado del buzón destino a su estado original antes del intento de copia.

Ejemplo:
```
C: A003 COPY 2:4 MEETING
S: A003 OK COPY completado
```

6.4.8. Comando UID

Argumentos: nombre de comando
argumentos del comando

Respuestas: OK comando UID completado

NO error en comando UID
BAD comando desconocido o argumentos no válidos

El comando UID tiene dos variantes. La primera, toma como argumento un comando COPY, FETCH, o STORE con los argumentos adecuados para cada comando. Sin embargo, los números del argumento conjunto de mensajes son identificadores únicos en lugar de números de secuencia de mensaje.

En la segunda variante, el comando UID recibe un comando SEARCH con los argumentos del propio comando. La interpretación de los argumentos es la misma que se hace con SEARCH; sin embargo, los números devueltos en una respuesta SEARCH para un comando UID SEARCH son identificadores únicos en lugar de números de secuencia de mensaje. Por ejemplo, el comando UID SEARCH 1:100 UID 443:557 devuelve los identificadores únicos correspondientes a la intersección del conjunto de números de secuencia de mensaje 1:100 y el conjunto de UID 443:557.

Se permiten rangos de conjuntos de mensaje; sin embargo, no existe garantía de que los identificadores únicos sean contiguos. Cualquier identificador único no existente dentro de un rango de conjunto de mensajes es ignorado sin provocar ningún mensaje de error.

El número situado tras el "*" en una respuesta FETCH no etiquetada es siempre un número de secuencia de mensaje, no un identificador único, incluso para una respuesta del comando UID. Sin embargo, las implementaciones de servidor DEBEN incluir implícitamente el elemento de datos de mensaje UID como parte de cualquier respuesta FETCH originada por un comando UID, sin importar si un UID se concibió como un elemento de datos de mensaje para el FETCH.

Ejemplo: C: A999 UID FETCH 4827313:4828442 FLAGS
S: * 23 FECTH (FLAGS (\Seen) UID 4827313)
S: * 24 FETCH (FLAGS (\Seen) UID 4827943)
S: * 25 FETCH (FLAGS (\Seen) UID 4828442)
S: A999 UID FETCH completado

6.5. Comandos de Cliente - Experimental/En desarrollo

6.5.1. Comando X<átomo>

Argumentos: definidos por la implementación

Respuestas: definidos por la implementación

Resultados: OK comando completado
NO fallo
BAD comando desconocido o argumentos no válidos

Cualquier comando precedido de una X se trata de un comando experimental. Los comandos que no forman parte de esta especificación, de un estándar o revisión de un estándar de esta especificación, o de un protocolo experimental aprobado por IESG, DEBE hacer uso del prefijo X.

Cualquier respuesta añadida no etiquetada generada por un comando experimental DEBE venir precedida de una X. Las

implementaciones de servidor NO DEBEN enviar estas respuestas no etiquetadas, a menos que el cliente la solicite ejecutando el comando experimental asociado.

Ejemplo:

```
      C: a441 CAPABILITY
      S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4XPIG-LATIN
      S: a441 OK CAPABILITY completado
      C: A442 XPIG-LATIN
      S: * XPIG-LATIN ow-nay eaking-spay ig-pay atin-lay
      S: A442 OK XPIG-LATIN ompleted-cay
```

7. Respuestas de servidor

Hay tres tipos de respuestas de servidor: respuestas de estado, datos del servidor, y solicitud de continuación de comando. La información contenida en una respuesta de servidor, identificada por "Contenido:" en la descripción de la respuesta que se describe más abajo, viene descrita por su función, no por su sintaxis. La sintaxis exacta para las respuestas de servidor se describe en la sección Sintaxis Formal.

El cliente DEDE estar preparado para aceptar cualquier respuesta en cualquier momento.

Las respuestas de estado pueden ser tanto etiquetadas como no etiquetadas. Las respuestas etiquetadas indican el resultado de la ejecución de un comando de cliente (estado BAD, NO, o OK), y poseen una etiqueta que coincide con el comando.

Algunas respuestas de estado, y todas los datos de servidor, son no etiquetadas. Una respuesta no etiquetada se indica mediante el token "*" en lugar de una etiqueta. Las respuestas de estado no etiquetadas indican el reconocimiento por parte del servidor, o el estado del servidor que no indique la finalización de un comando (por

ejemplo, una alerta de apagado inminente del sistema). Por razones históricas, las respuestas de datos de servidor no etiquetadas se denominan también "datos no solicitados", aunque en realidad solo las respuestas unilaterales de datos de servidor son realmente "no solicitadas".

Ciertos datos de servidor DEBEN ser almacenados por el cliente cuando se reciben; estos datos son reconocidos mediante su descripción. Estos datos constituyen información crítica que afecta a la interpretación de todos los comandos y respuestas posteriores (p.ej. actualizaciones que reflejan la creación o destrucción de mensajes).

Otros datos de servidor DEBERIAN ser almacenados para

posteriores consultas; si el cliente no necesita grabar los datos, o si la grabación de estos datos no tiene un propósito claro (p.ej. una respuesta SEARCH sin comando SEARCH en progreso), los datos DEBERIAN ser ignorados.

Un ejemplo de datos de servidor no etiquetados unilaterales sucede cuando la conexión IMAP se encuentra en estado seleccionado. En el estado seleccionado, el servidor comprueba si hay nuevos mensajes en el buzón como parte de la ejecución del comando. Normalmente, esto se hace en la ejecución de cualquier comando; por tanto, un comando NOOP es suficiente para comprobar la existencia de nuevosmensajes. Si hay mensajes nuevos, el servidor envía respuestas no etiquetadas EXISTS y RECENT indicando el nuevo tamaño del buzón. Las implementaciones de servidor que ofrecen múltiples accesos simultáneos al mismo buzón DEBERIAN también enviar las respuestas adecuadas FETCH y EXPUNGE de forma unilateral si cualquier otro agente cambia el estado de las banderas de los mensajes o elimina cualquier mensaje.

Las respuestas de solicitud de continuación de comando utilizan el token "+" en lugar de una etiqueta. El servidor envía estas respuestas para indicar la aceptación de un comando de cliente incompleto y la disponibilidad del recordatorio del comando.

7.1. Respuestas de servidor - Respuestas de estado

Las respuestas de estado son OK, NO, BAD, PREAUTH y BYE. Las respuestas OK, BAD y NO pueden estar etiquetadas o no. PREAUTH y BYE van siempre etiquetadas.

Las respuestas de estado ES POSIBLE que incluyan un "código de respuesta" OPCIONAL. Un código de respuesta consiste en datos encerrados entre corchetes en forma de un átomo, seguidos posiblemente de un espacio y argumentos. El código de respuesta contiene información adicional o códigos de estado para el software del cliente más allá de la condición OK/NO/BAD, y se definen cuando un cliente lleva a cabo una acción especifica en base a dicha información adicional.

Los códigos de respuesta definidos actualmente son:

ALERT — El texto inteligible por el usuario contiene una alerta especial que DEBE mostrarse al usuario de manera que el mensaje consiga atraer la atención del mismo.

NEWNAME — Seguido de un nombre de buzón y un nombre de nuevo buzón. Un SELECT o EXAMINE no se ejecutan correctamente porque el nombre de buzón de destino no existe porque ha sido renombrado con el nuevo nombre de buzón. Esto indica al usuario que la operación puede tener éxito si se vuelven a ejecutar los comandos SELECT, EXAMINE con el nuevo nombre de buzón.

PARSE — El texto inteligible por el usuario representa un error en el procesado de la

cabecera [RFC-822] o cabeceras [MIME-IMB] de un mensaje del buzón.

PERMANENTFLAGS Seguido de una lista entre paréntesis de banderas, indica al cliente cual de las banderas conocidaspueden ser cambiadas de manera permanente. Cualquier bandera presente en la respuesta no etiquetada FLAGS,pero no en la lista PERMANENTFLAGS, no puede ser establecida de forma permanente. Si el cliente intenta ejecutar un STORE de una bandera que no se encuentra en la lista PERMANENTFLAGS, el servidor lorechazará con una respuesta NO o almacenará el estado del recordatorio de la sesión

actual. La lista PERMANENTFLAGS también puede incluir la bandera especial *, que indica que es posible crear nuevas claves mediante el almacenamiento de dichas banderas en el buzón.

READ-ONLY El buzón está seleccionado en modo solo lectura, o durante su selección su estado a cambiado de lectura-escritura a solo lectura.

READ-WRITE El buzón está seleccionado en modo lectura-escritura, o durante su selección su estado a cambiado de solo lectura a lectura-escritura.

TRYCREATE Comandos como APPEND o COPY fallan en su ejecución porque el buzón destino no existe (a diferencia de cualquier otra razón). Esto indica al cliente que la operación puede tener éxito si se ejecuta primero el comando CREATE para crear el buzón.

UIDVALIDITY Seguido por un número decimal, indica el valor de vigencia de identificador único.

UNSEEN Seguido por un número decimal, indica el número del primer mensaje cuya bandera \Seen no está activada.

Otros códigos de respuesta definidos por implementaciones concretas de cliente o servidor DEBERIAN venir precedidos de una "X" hasta que sean incluidos en un a revisión de este protocolo.

7.1.1. Respuesta OK

Contenido: código de respuesta OPCIONAL
texto inteligible por el usuario

La respuesta OK representa un mensaje informativo del servidor. Cuando esta está etiquetada, indica que el comando asociado ha sido ejecutado con éxito. El texto inteligible por el usuario ES POSIBLE que sea presentado al usuario como un mensaje informativo. La forma no etiquetada indica un mensaje de contenido meramente informativo; la naturaleza de la información ES POSIBLE

que se indique mediante un código de respuesta.

La forma no etiquetada también se utiliza como una de las tres posibles formas de reconocimiento que se realizan al comienzo de cada conexión. Indica que la conexión no está autentificada todavía y es preciso un comando LOGIN.

Ejemplo:

```
S: * OK IMAP4rev1 servidor preparado
C: A001 LOGIN fred blurdyboop
S: * OK [ALERT] apagado del sistema en 10 minutos
S: A001 OK LOGIN completado
```

7.1.2. Respuesta NO

Contenido: código de respuesta OPCIONAL
texto inteligible por el usuario

La respuesta NO indica un mensaje de error operacional del servidor.Cuando está etiquetada, indica que el comando asociado se ha ejecutado correctamente. Si no está etiquetada indica una advertencia; el comando puede completarse con éxito. El texto inteligible por el usuario describe la condición.

Ejemplo:
```
C: A222 COPY 1:2 owatagusiam
S: * NO 98% del disco lleno, por favor
borre datos innecesarios
S: A222 OK COPY completado
C: A223 COPY 3:200 blurdyboop
S: * NO 98% del disco lleno, por favor
borre datos innecesarios
S: * NO 98% del disco lleno, por favor
borre datos innecesarios
S: A223 NO COPY fallido: disco lleno
```

7.1.3. Respuesta BAD

Contenidos: código de respuesta OPCIONAL
texto inteligible por el usuario

La respuesta BAD indica un mensaje de error del servidor. Cuando está etiquetada, informa de un error a nivel de protocolo en el comando de cliente; la etiqueta indica el

comando que causó el error. La forma no etiquetada indica un error a nivel de protocolo en el cual no se reconoce el comando asociado; también puede indicar un fallo interno en el servidor. El texto inteligible por el usuario describe la condición.

Ejemplo:
```
C: ...línea de comando muy larga
S: * BAD línea de comando demasiado larga
C: ...línea vacía...
S: * BAD línea de comando vacía
```

```
                  C: A443 EXPUNGE
                  S: * BAD fallo en disco, intentando
                  recuperación a un disco nuevo!
                  S: * OK recuperación con éxito, sin pérdida
                  de datos
                  S: A443 OK borrado completado
```

7.1.4. Respuesta PREAUTH

Contenido: código de respuesta OPCIONAL
texto inteligible por el usuario

La respuesta PREAUTH está siempre etiquetada, y es una de las tres posibles formas de reconocimiento que se realizan al comienzo de cada conexión. Esto indica que la conexión ya está autentificada por medios externos y por tanto no es necesario ejecutar un comando LOGIN.

Ejemplo: S: * PREAUTH servidor IMAP4rev1 identificado en el sistema como Smith

7.1.5. Respuesta BYE

Contenido: código de respuesta OPCIONAL
texto inteligible por el usuario

La respuesta BYE siempre posee forma no etiquetada, e indica que el servidor está en vías de cerrar la conexión. El texto inteligible por el usuario ES POSIBLE que se muestre al usuario como informe de estado por parte cliente. La respuesta BYE se envía en base a una de estas cuatro condiciones:

1) como parte de una secuencia logout normal. El servidor cerrará la conexión tras el envío de la respuesta etiquetada OK al comando LOGOUT.

2) como anuncio de un apagado inmediato del sistema. El servidor cierra la conexión inmediatamente.

3) como anuncio de un autologout por inactividad. El servidor cierra la conexión inmediatamente.

4) como una de las tres formas posibles de reconocimiento al comienzo de la conexión, indicando que el servidor no va a permitir una conexión por parte del cliente en cuestión. El servidor cierra la conexión inmediatamente.

La diferencia entre un BYE que tiene lugar como parte de una secuencia LOGOUT normal (el primer caso) y un BYE que tiene lugar a causa de un fallo (los otros tres casos) es que la conexión se cierra inmediatamente en el caso de que haya fallos.

Ejemplo:

```
S: * BYE Autologout; demasiado tiempo inactivo
```

7.2. Respuestas de servidor - Estados del servidor y del buzón

Estas respuestas son siempre no etiquetadas. Explican la forma en que los datos de estado del servidor y buzón se

transmiten desde el servidor al cliente. Muchas de estas respuestas se producen como resultado de un comando del mismo nombre.

7.2.1. Respuesta CAPABILITY

in 3
Contenido: lista de aptitudes

La respuesta CAPABILITY tiene lugar como resultado de un comando CAPABILITY. La lista de capacidades contiene una lista de nombres de aptitudes separadas por espacios que el servidor soporta. La lista de capacidades DEBE incluir el átomo "IMAP4rev1".

Un nombre de capacidad que comience con "AUTH=" indica que el servidor soporta ese mecanismo de autentificación especial.

Otros nombres de aptitud indican que el servidor soporta unaextensión, revisión, o corrección del protocolo

IMAP4rev1. Las respuestas de servidor DEBEN ser conformes a este documento hasta que el cliente elabore un comando que utilice la aptitud asociada.

Los nombres de aptitudes DEBEN o comenzar con "X" o ser extensiones, revisiones o rectificaciones de IMAP4rev1 estándar o en vías de estandarización registrados por la IANA. Un servidor NO DEBE ofrecer nombres de aptitudes no registradas o no estándar, a menos que tales nombres vengan precedidos por "X".

Las implementaciones de cliente NO DEBERIAN necesitar un nombre de aptitud que no sea "IMAP4rev1", y DEBE ignorar cualquier nombre de aptitud desconocido.

Ejemplo:

S: * CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 XPIG-LATIN

7.2.2. Respuesta LIST

Contenido: atributos de nombre
delimitador de jerarquía
nombre

La respuesta LIST sucede como resultado de un comando LIST. Devuelve un único nombre que coincida con lo especificado en LIST. Puede haber múltiples respuestas LIST para un único comando LIST.

Se definen cuatro atributos de nombre:

\Noinferiors Los niveles hijos de la jerarquía no pueden salir al exterior con este nombre; no existe ningún nivel hijo en este momento y no pueden ser creados en un futuro.

\Noselect No puede utilizarse este nombre como buzón seleccionable.

\Marked El buzón ha sido catalogado como "interesante" por el servidor; el buzón probablemente contiene mensajes añadidos

desde la última vez que el buzón fue seleccionado.

\Unmarked El buzón no contiene ningún mensaje

adicional desde la última vez que fue seleccionado.

Si no es factible para el servidor determinar si el buzón es o no "interesante", o si el nombre es un nombre \Noselect, el servidor NO DEBERIA enviar \Marked o \Unmarked.

El delimitador de jerarquía es un carácter que se utiliza paradelimitar los niveles de la jerarquía de un buzón. Un cliente puede hacer uso de este carácter para crear buzones hijo, y parabuscar en niveles más altos o más bajos en la jerarquía denombres. Todo hijo de un nodo de alto nivel en la jerarquía DEBE usar el mismo carácter separador. Un delimitador de jerarquía NIL significa que no existe jerarquía; el nombre es un nombre "plano".

El nombre representa una jerarquía de izquierda a derecha no ambigua, y DEBE poder ser utilizado como referencia en los comandos LIST y LSUB. A menos que se indique \Noselect, el nombre DEBE ser válido para ser utilizado como argumento en los comandos, tales como SELECT, que acepta nombres de buzón.

Ejemplo: S: * LIST (\Noselect) "/" ~/Mail/foo

7.2.3. Respuesta LSUB

Contenido: atributos de nombre
delimitador de jerarquía
nombre

La respuesta LSUB tiene lugar como resultado de un comando LSUB. Devuelve un único nombre que coincide con lo especificado en LSUB. Puede haber múltiples respuestas LSUB para un único comando LSUB. El formato de los datos es idéntico al definido en la respuesta LIST.

Ejemplo: S: * LSUB "." #news.comp.mil.misc

7.2.4. Respuesta STATUS

Contenido: nombre
lista de estados entre paréntesis

La respuesta STATUS tiene lugar como resultado de un comando STATUS. Devuelve el nombre de buzón que coincida con la especificación dada en STATUS y la información

de estado del buzón solicitado.

Ejemplo:
S: * STATUS blurdybloop (Mensajes 231 UIDNEXT 44292)

7.2.5. Respuesta SEARCH

Contenido: cero o más números

La respuesta SEARCH tiene lugar como resultado de un comando SEARCH o UID SEARCH. El/los número/s se refieren a aquellos mensajes que coincidan con el patrón de búsqueda dado. Para SEARCH, estos son números de secuencia de mensaje; para UID SEARCH, estos son identificadores únicos. Cada número está delimitado mediante un espacio.

Ejemplo: S: * SEARCH 2 3 6

7.2.6. Respuesta FLAGS

Contenido: lista de banderas entre paréntesis

La respuesta FLAGS tiene lugar como resultado de un comando SELECT o EXAMINE. La lista de banderas entre paréntesis identifica las banderas (como mínimo, las banderas definidas por el sistema) que son aplicables a este buzón. Puede haber tambiénbanderas no definidas por el sistema, en función de la implementación de servidor.

La actualización desde la respuesta FLAGS DEBE ser almacenada por el cliente.

Ejemplo:
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)

7.3. Respuestas de servidor - Tamaño del buzón

Son respuestas siempre no etiquetadas. Esto explica el modo en que los cambios en el tamaño del buzón se transmiten del servidor al cliente. A continuación del token "*" sigue un número que representa un contador de mensajes.

7.3.1. Respuesta EXISTS

Contenido: ninguno

La respuesta EXISTS informa acerca del número de mensajes contenidos en el buzón. Esta respuesta tiene lugar como resultado de un comando SELECT o EXAMINE, y si el tamaño del buzón cambia (p.ej. nuevo mensaje).

La actualización desde la respuesta EXISTS DEBE ser almacenada por el cliente.

Ejemplo: S: * 23 EXISTS

7.3.2. Respuesta RECENT

Contenido: ninguno

La respuesta RECENT informa acerca del número de mensajes cuya bandera \Recent está activada. Esta respuesta tiene lugar como resultado de un comando SELECT o EXAMINE, y si el tamaño del buzón cambia (p.ej. nuevo mensaje).

Nota: No es seguro que los números de secuencia de mensaje de los mensajes recientes sean un rango contiguo de los n mensajes más altos en el buzón

(donde n es el valor devuelto por la respuesta RECENT). Ejemplos de situaciones en las que esto no sucede son: múltiples clientes con el mismo buzón abierto (en la primera sesión el mensaje será notificado como reciente, en sesiones posteriores se verá como no reciente), y cuando el buzón es reordenado por un agente no IMAP.

La única forma fiable de identificar los mensajes recientes es mirar las banderas de mensaje para comprobar cual de ellos tiene la bandera \Recent activada, o hacer un SEARCH RECENT.

La actualización desde la respuesta RECENT DEBE ser almacenada por el cliente.

Ejemplo: S: * 5 RECENT

7.4. Respuestas de servidor - Estado del mensaje

Estas respuestas son siempre no etiquetadas. Esto explica la forma en la que los datos del mensaje de transmiten desde el servidor al cliente, a menudo como resultado de un comando del mismo nombre. A continuación del token "*"

sigue un número que representa un número de secuencia de mensaje.

7.4.1. Respuesta EXPUNGE

Contenido: ninguno

La respuesta EXPUNGE informa de que el número de secuencia de mensaje especificado ha sido eliminado de manera permanente del buzón. El número de secuencia de mensaje para cada mensaje en la sucesión se ve decrementado en 1, y esto se refleja en el número de secuencia de mensaje de las respuestas sucesivas (incluyendo otras respuestas EXPUNGE no etiquetadas).

Como resultado de esta norma de reducción, los números de secuencia de mensaje que aparecen en un conjunto de respuestas EXPUNGE sucesivas dependen de si los mensajes se eliminan comenzando desde los números más bajos a los más altos, o viceversa. Por ejemplo, si los últimos 5 mensajes en un buzón que contenga 9 mensajes, son eliminados; un servidor "más bajo a más alto" enviará cinco respuestas EXPUNGE no etiquetadas para el número de secuencia de mensaje 5, mientras que un servidor "más alto a más bajo" enviará respuestas no etiquetadas EXPUNGE para los números de secuencia de mensaje 9,8,7,6 y 5.

Una respuesta EXPUNGE NO DEBE enviarse cuando no hay ningún comando en progreso; ni en el proceso de contestación de un comando FETCH, STORE o SEARCH. Esta regla es necesaria para evitar una pérdida de sincronización de los números de secuencia de mensaje entre cliente y servidor.

La actualización desde la respuesta EXPUNGE DEBE ser almacenada por el cliente.

Ejemplo: S: * 44 EXPUNGE

7.4.2. Respuesta FETCH

Contenido: datos del mensaje

La respuesta FETCH devuelve datos acerca de un mensaje al cliente. Los datos son pares de nombres de elementos de datos y sus valores entre paréntesis. Esta respuesta tiene lugar como resultado de un comando FETCH o STORE,

o como resultado de una decisión unilateral por parte del servidor (p.ej actualizaciones de banderas).

Los elementos de datos actuales son:

BODY — Forma de BODYSTRUCTURE sin datos de extensión

BODY[<sección>]<<octeto-origen>> — Cadena que representa el contenido del cuerpo de la sección especificada. La cadena DEBERIA ser interpretada por el cliente en función de la codificación de transferencia de contenido, tipo de cuerpo, y subtipo.

Si se especifica el octeto origen, esta cadena es una subcadena del contenido total del cuerpo, que comienza con ese octeto origen. Esto significa que BODY[]<0> ES POSIBLE que sea truncado, pero BODY[] NUNCA será truncado.

Se permiten datos tipo texto de 8-bit si un identificador de [CHARSET] forma parte de la lista entre paréntesis de parámetros de cuerpo de esta sección. Hay que tener en cuenta que las cabeceras (especificadores de parte HEADER o MIME, o la porción de cabecera de una parte MESSAGE/RFC822), DEBEN ser representadas por 7-bit; no se permiten caracteres de 8-bit en las cabeceras. También hay que reseñar que la línea en blanco situada al final de la cabecera siempre se incluye en los datos de cabecera.

Los datos no tipo texto tales como los datos binarios DEBEN ser codificados en transferencia en formato texto tales como BASE64 antes de ser enviados al cliente. Para volver a los datos binarios originales, el cliente DEBE decodificar la cadena codificada en transferencia.

BODYSTRUCTURE — Lista entre paréntesis que describe la estructura del cuerpo de un mensaje [MIME-IMB]. Esto es analizado por el

servidor procesando los campos de cabecera [MIME-IMB], dejando tantos campos como sea necesario.

Por ejemplo, un mensaje de texto de 48 líneas y 2279 octetos puede tener la siguiente estructura de cuerpo: ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48)

Si tiene múltiples partes se utiliza anidación mediante paréntesis. En lugar de tener un tipo de cuerpo como primer elemento de la lista entre paréntesis tenemos un cuerpo anidado. El segundo elemento de la lista entre paréntesis es el subtipo de la multiparte (mixto, resumen, paralelo, alternativo,etc.).

Por ejemplo, un mensaje compuesto de dos partes, un texto y un texto codificado en BASE645 adjunto, puede tener la siguiente estructura de cuerpo: (("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23) ("TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME" "cc.diff") "<960723163407.20117@cac.washington.edu>" "Compilador diff" "BASE64" 4554 73) "MIXTO"))

Los datos de extensión van a continuación del subtipo multiparte. Nunca se devuelven datos de extensión con la búsqueda BODY, pero si pueden devolverse con una búsqueda BODYSTRUCTURE. Los datos de extensión, si existen, DEBEN situarse en el orden definido.

Los datos de extensión de una parte de cuerpo multiparte se encuentran en el orden siguiente:

lista entre paréntesisde parámetros de cuerpo
: Una lista entre paréntesis de pares atributo-valor [p.ej. ("foo" "bar" "baz" "rag") donde "bar" el valor de

"foo" y "rag"es el valor de "baz"] como se define en [MIME-IMB].

localización del cuerpo
: Una lista entre paréntesis, consistente en una cadena de tipo localización seguido de una lista entre paréntesis de pares atributo/valor de tipo localización. El tipo localización y los nombres de atributos serán definidos en una futura revisión en vías de estandarización de [DISPOSITION].

idioma del cuerpo
: Una cadena o lista entre paréntesis que especifica el valor para el

idioma del cuerpo como se define en [LANGUAGE-TAGS].

Cualquier dato de extensión adicional no está definido hasta el momento en esta versión del protocolo. Estos datos de extensión pueden consistir en cero o más NIL, cadenas, números, o listas entre paréntesis anidadas de tales datos. Las implementaciones de cliente que hagan una búsqueda BODYSTRUCTURE DEBEN estar preparados para aceptar este tipo de datos de extensión. Las implementaciones de servidor NO DEBEN enviar ese tipo de datos de extensión hasta que una nueva revisión de este protocolo los defina.

Los campos básicos de una parte de cuerpo no multiparte se encuentran en el orden siguiente:

tipo de cuerpo
Una cadena que especifica el nombre del tipo de contenido como se define en [MIME-IMB].

subtipo del cuerpo
Una cadena que especifica el nombre del subtipo de contenido como se

define en [MIME-IMB].

lista entre paréntesis de parámetros de cuerpo
Una lista entre paréntesis de pares atributo/valor [p.ej. ("foo" "bar" "baz" "rag") donde "bar" es el valor de "foo" y "rag" es el valor de "baz"] como se define en [MIME-IMB].

identificador de cuerpo
Una cadena que especifica el identificador de contenido como se define en [MIME-IMB].

descripción del cuerpo
Una cadena que especifica la descripción del contenido como se define en [MIME-IMB].

codificación del cuerpo
Una cadena que especifica la codificación en transferencia del contenido como se define en [MIME-IMB].

tamaño del cuerpo
Un número que especifica el tamaño del cuerpo en octetos. Hay que tener en cuenta que este tamaño es el tamaño de su codificación en transferencia y no el tamaño total tras la decodificación.

Un tipo de cuerpo de tipo MESSAGE y subtipo RFC822 contiene, inmediatamente después de los campos básicos, la estructura de envoltorio, estructura del cuerpo, y el tamaño en líneas de texto del mensaje encapsulado.

Un tipo de cuerpo de tipo TEXT contiene, inmediatamente después de los campos básicos, el tamaño del cuerpo en líneas de texto. Hay que tener en cuenta que este tamaño es el

tamaño de su codificación en transferencia y no el tamaño total tras la decodificación.

Los datos de extensión siguen los campos básicos y los campos de tipo especifico mostrados más abajo. Nunca se devuelven datos de extensión con la búsqueda BODY, pero si pueden devolverse con una búsqueda BODYSTRUCTURE. Los datos extensibles, si existen, DEBEN situarse en el orden definido.

Los datos de extensión de una parte de cuerpo no multiparte están en el orden siguiente:

cuerpo MD5
: Una cadena que especifica el valor del cuerpo MD5 como se define en [MD5].

disposición del cuerpo
: Una lista entre paréntesis con el mismo contenido y función que la disposición del cuerpo de una parte de cuerpo multiparte.

idioma del cuerpo
: Una cadena o lista entre paréntesis que indica el valor para el idioma como se definen en [LANGUAGE-TAGS].

Cualquier dato de extensión adicional no está definido hasta el momento en esta versión del protocolo, y seguiría la pauta definida más abajo para los datos de extensión multiparte.

ENVELOPE
: Una lista entre paréntesis que describe la estructura de envoltorio de un mensaje. Esto es analizado por el servidor procesando la cabecera [RFC-822] en partes de componente, dejando los campos necesarios.

Los campos de la estructura de envoltorio están en el orden siguiente: fecha, asunto, desde, remite, remite a, respuesta a, a, cc, bcc, en respuesta a, e identificador de mensaje. Los campos fecha, asunto, en respuesta a, e identificador de mensaje son cadenas. Los campos desde, remite, respuesta a, a, cc, y bcc son listas entre paréntesis de estructuras de dirección.

Una estructura de dirección es una lista entreparéntesis que describe una dirección de correo electrónico. Los campos de una estructura de dirección están en el orden siguiente: nombre personal, [SMTP] en la lista de dominio (ruta origen), nombre de buzón, y nombre de host.

La sintaxis de grupo [RFC-822] se indica mediante una forma especial de estructura de dirección en la cual el nombre de host es NIL. Si el campo de nombre de buzón es también NIL, esto es un fin de marcador de grupo (punto y coma en la sintaxis RFC-822). Si el campo de nombre de buzón no es NIL, esto es un comienzo de marcador de grupo, y el campo de nombre de buzón contiene la expresión de nombre de grupo.

Cualquier campo de un envoltorio o estructura de dirección no aplicable se especifica mediante NIL. Hay que tener en cuenta que el servidor DEBE averiguar los campos respuesta a y remite a partir del campo desde; un cliente no tiene por que saber hacerlo.

FLAGS Una lista entre paréntesis de banderas que están activas para este mensaje.

INTERNALDATE Una cadena que representa la fecha interna delmensaje.

RFC822 Equivalente a BODY[].

RFC822.HEADER Equivalente a BODY.PEEK[HEADER].

RFC822.SIZE Un número que expresa el tamaño [RFC-822] del mensaje.

RFC822.TEXT Equivalente a BODY[TEXT].

UID Un número que expresa el identificador único del mensaje.

Ejemplo: S: * 23 FETCH (FLAGS (\Seen) RFC822.SIZE 44827)

7.5. Respuestas de servidor - Solicitud de continuación de comando

La respuesta de solicitud de continuación de comando se indica mediante un token "+"en lugar de una etiqueta. Esta forma de respuesta indica que el servidor está preparado para aceptar la continuación de un comando desde el cliente. El recordatorio de esta respuesta es una línea de texto.

Esta respuesta se utiliza en el comando AUTHORIZATION para trasmitir datos de servidor al cliente, y solicitar datos del cliente adicionales. Esta respuesta también se usa si un argumento de cualquier comando es un literal.

El cliente no puede enviar los octetos del literal a menos que el servidor indique que los espera. Esto permite al servidor procesar comandos y rechazar errores en modo línea a línea. El recordatorio del comando, incluyendo el CRLF que finaliza el comando, sigue a los octetos del literal. Si hay argumentos de comando adicionales, a continuación de los octetos del literal viene un espacio y estos argumentos.

Ejemplo:

```
C: A001 LOGIN {11}
S: + Preparado para texto de comando adicional
C: FRED FOOBAR {7}
S: + Preparado para texto de comando adicional
C: hombre gordo
S: A001 OK LOGIN completado
C: A044 BLURDYBLOOP {102856}
S: A044 BAD no hay comando "BLURDYBLOOP"
```


8. Ejemplo de conexión IMAP4rev1

Las siguientes líneas constituyen una transcripción de una conexión IMAP4rev1. Las líneas largas se fraccionan por claridad en el ejemplo.

```
S: * OK IMAP4rev1 servicio dispuesto
C: a001 login mrc secreto
S: a001 OK LOGIN completado
C: a002 select inboxS: * 18 EXISTS
S: * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S: * 2 RECENT
S: * OK [UNSEEN 17] Mensaje 17 es el primer mensaje no leído
S: * OK [UIDVALIDITY 3857529045] UIDs válido

S: a002 OK [READ-WRITE] SELECT completado
C: a003 fetch 12 full
S: * 12 fetch [FLAGS (\Seen) INTERNALDATE "17-Jul-1996
   02:44:25 -0700" RFC-822.SIZE 4286 ENVELOPE ("Wed,
   17 Jul 1996 02:23:25 -0700 (PDT)" "IMAP4rev1 WG mtg
   resumen y minutos"
   (("Terry Gray" NIL "gray" "cac.washington.edu"))
   (("Terry Gray" NIL "gray" "cac.washington.edu"))
   (("Terry Gray" NIL "gray" "cac.washington.edu"))
   ((NIL NIL "imap" "cac.washington.edu"))
   ((NIL NIL "minutos" "CNRI.Reston.VA.US")
   ("John Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
   "<B27397-0100000@cac.washington.edu>")
   BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT"
```

```
      3028 92))
   S: a003 OK FETCH completado
   C: a004 búsqueda 12 cuerpo[cabecera]
   S: * 12 FETCH (BODY[HEADER] {350}
   S: Fecha: Wed, 17 Jul 1996 02:23:25 0700 (PDT)
   S: Desde: Terry Gray <gray@cac.washington.edu>
   S: Asunto: IMAP4rev1 WG mtg resumen y minutos
   S: A: imap@cac.washington.edu
   S: cc: minutes@CNRI.Reston.VA.US , John Klensin
      <KLENSIN@INFOOODS.MIT.EDU> .....
   S: Identificador de mensaje:
      <B27397-0100000@cac.washington.edu>
   S: Version MIME: 1.0
   S: Tipo de contenido: TEXT/PLAIN; CHARSET=US-ASCII
   S:
   S: )
   S: a004 OK FETCH completado
   C: a005 almacenar 12 +banderas \borrado
   S: * 12 FETCH (FLAGS (\Seen \Deleted))
```



```
   S: a005 OK +FLAGS completadoC: a006 desconexión
   S: * BYE IMAP4rev1 servidor finalizando conexión
   S: a006 OK LOGOUT completado
```

9. Sintaxis formal

La siguiente especificación de sintaxis hace uso de la notación Backus-Naur Form (BNF) aumentada como se especifica en [RFC-822] con una excepción; el delimitador usado con el constructor "#" es un espacio (ESPACIO) y no una o más comas.

En caso de reglas alternativas u opcionales en el cual una regla posterior solape una regla anterior, la regla listada antes en el tiempo DEBE tener prioridad. Por ejemplo, "\Seen" cuando se analiza como bandera es el nombre de bandera \Seen y no una extensión de bandera, aunque "\Seen" podría ser procesado como una extensión_bandera. Alguna, pero no todas las instancias de esta regla se reflejan más abajo.

A excepción de los especificados, todos los caracteres son no sensibles a mayúsculas-minúsculas. El uso de mayúsculas-minúsculas utilizados para definir cadenas de token se hace por claridad editorial. Las implementaciones DEBEN aceptar estas cadenas en modo no sensible a mayúsculas-minúsculas.

```
dirección       ::= "(" dirección_nombre ESPACIO dirección_adl
                    ESPACIO dirección_buzón_de_correo ESPACIO
                    dirección_host ")"

dirección_adl   ::= cadenan
                    ;; contiene la ruta desde la dirección de
                    ;; ruta [RFC-822] si es no NIL

dirección_host  ::= cadenan
                    ;; NIL indica sintaxis de grupo
                    ;; [RFC-822]. Si no, contiene el
                    ;; nombre de dominio [RFC-822]

dirección_buzón_de_correo       ::= cadenan
                                    ;; NIL indica fin de grupo
                                    ;; [RFC-822]; si es no NIL y
```

```
                                    ;; dirección_host es NIL,
                                    ;; contiene el nombre de
```



```
                                    ;; de grupo [RFC-822]
                                    ;; En cualquier otro caso,
                                    ;; contiene parte-local
                                    ;; [RFC-822]

dirección_nombre        ::=     cadena_n
                                ;; contiene la expresión del
                                ;; buzón [RFC-822] si es no NIL

alfabético      ::=     "A" / "B" / "C" / "D" / "E" / "F" / "G" /
                        "H" / "I" / "J" / "K" / "L" / "M" / "N" /
                        "O" / "P" / "Q" / "R" / "S" / "T" / "U" /
                        "V" / "W" / "X" / "Y" / "Z" / "a" / "b" /
                        "c" / "d" / "e" / "f" / "g" / "h" / "i" /
                        "j" / "k" / "l" / "m" / "n" / "o" / "p" /
                        "q" / "r" / "s" / "t" / "u" / "v" / "w" /
                        "x" / "y" / "z"
                        ;; sensible a mayúsculas-minúsculas

añadir  ::= "APPEND" ESPACIO buzón [ESPACIO lista_banderas]
            [ESPACIO fecha_hora] ESPACIO literal

acadena ::=  átomo / cadena

átomo   ::=     1*ÁTOMO_CARÁCTER

ÁTOMO_CARÁCTER  ::= <cualquier carácter excepto átomos_especiales>

átomos_especiales       ::= "(" / ")" / "{" / ESPACIO / CTL /
                            lista_comodines /
                             especiales_ent_comillas



autentificar    ::= "AUTHENTICATE" ESPACIO tipo_auth *(CRLF base64)

tipo_auth       ::= átomo
                    ;; definido en [IMAP-AUTH]

base64  ::= *(4caracteres_base64) [terminal_base64]

caracteres_base64       ::= alfabético / dígito / "+" / "/"

terminal_base64 ::= (2caracteres_base64 "==") /
                    (3caracteres_base64 "=")

cuerpo  ::= "(" tipo_parte1_cuerpo / tipo_partem_cuerpo ")"
```



```
extensión_cuerpo        ::= cadenan / número /
                            "(" 1#extensión_cuerpo ")"
                            ;; Expansión futura. Las
                            ;; implementaciones de cliente
                            ;; DEBEN aceptar campos de
                            ;; extensión_cuerpo. Las
                            ;; implementaciones de servidor
```

```
                            ;; NO DEBEN generar campos de
                            ;; extensión_de_cuerpo excepto
                            ;; aquellos definidos en futuras
                           ;; revisiones estándar o en vías
                            ;; de estandarización de esta
                            ;; especificación.

ext_cuerpo_parte1       ::= campo_md5_cuerpo
                            [ESPACIO campo_disp_cuerpo
                            [ESPACIO campo_lang_cuerpo
                            [ESPACIO 1#extensión_cuerpo]]]
                            ;; NO DEBE devolverse en
                            ;; búsquedas "BODY" no
                            ;; extensibles

ext_cuerpo_partem       ::= campo_param_cuerpo
                            [ESPACIO campo_disp_cuerpo
                            ESPACIO campo_lang_cuerpo
                            [ESPACIO 1#extensión_cuerpo]]
                           ;; NO DEBE devolverse en
                            ;; búsquedas "BODY" no
                            ;; extensibles

campos_cuerpo   ::= campo_param_cuerpo ESPACIO campo_id_cuerpo
                    ESPACIO campo_desc_cuerpo ESPACIO
                     campo_enc_cuerpo ESPACIO
                     campo_octetos_cuerpo

campo_desc_cuerpo       ::= cadenan

campo_disp_cuerpo       ::="(" cadena ESPACIO
                            campo_param_cuerpo ")"/ nil

campo_enc_cuerpo        ::= (<"> ("7BIT" / "8BIT" / "BINARY" /
                            "BASE64" /
                            "imprimible_ent_comillas")
                            <">) / cadena

campo_id_cuerpo         ::= cadenan

campo_lang_cuerpo       ::= cadenan / "(" 1#cadena ")"
```



```
campo_líneas_cuerpo     ::= número

campo_md5_cuerpo        ::= cadenan

campo_octetos_cuerpo    ::= número

campo_param_cuerpo      ::= "(" 1#(cadena ESPACIO cadena ")"
                            / nil

tipo_parte1_cuerpo      ::= (tipo_básico_cuerpo /
                            tipo_mensg_cuerpo
                            / tipo_texto_cuerpo)
                            [ESPACIO ext_cuerpo_parte1]

tipo_básico_cuerpo      ::= ESPACIO campos_cuerpo
                            ;; subtipo MESSAGE NO DEBE
                            ;; ser "RFC822"

tipo_partem_cuerpo      ::= 1*cuerpo ESPACIO subtipo_multimedia
                            [ESPACIO ext_cuerpo_partem]

tipo_mensg_cuerpo       ::= mensaje_multimedia ESPACIO
                            campos_cuerpo
                            ESPACIO envoltorio ESPACIO cuerpo
```

```
                            ESPACIO campo_líneas_cuerpo

tipo_texto_cuerpo       ::= texto_multimedia ESPACIO
                            campos_cuerpo
                            ESPACIO campo_líneas_cuerpo

aptitud         ::="AUTH=" tipo_auth / átomo
                    ;; Las aptitudes nuevas DEBEN comenzar con
                   ;; "X" o ser registradas por la IANA como
                    ;; estándar o en vías de estandarización

datos_aptitud   ::= "CAPABILITY" ESPACIO [1#aptitud ESPACIO
                    "IMAP4rev1"
                    [ESPACIO 1#funcionalidad]
                    ;; Los servidores IMAP4rev1 que ofrezcan
                   ;; compatibilidad RFC 1730 DEBEN incluir
                   ;; como primera aptitud "IMAP4"

CARACTER        ::= <cualquier carácter US-ASCII de 7-bit
                    excepto NUL, 0x01 - 0x7f>

CARACTER8       ::= <cualquier octeto de 8-bit excepto NUL,
                    0x01 - 0xff>
```



```
comando ::= etiqueta ESPACIO (cualquier_comando /
            comando_auth / comando_noauth / comando_select)
            CRLF
            ;; Modo basado en el estado

cualquier_comando       ::= "CAPABILITY" / "LOGOUT" / "NOOP" /
                            comando_x
                            ;; válidos en todos los estados

comando_auth    ::= añadir / crear / borrar / examinar /
                    listar /lsub / renombrar / seleccionar /
                    estado / suscribir / abandonar_suscripción
                    ;; válidos sólo en los estados
                    ;; autentificado y seleccionado

comando_no_auth ::= entrar / autentificar
                            ;; válidos sólo en el estado no
                            ;; autentificado

comando_select  ::= "CHECK" / "CLOSE" / "EXPUNGE" / copiar /
                   recuperar / almacenar / uid / buscar
                    ;; válidos sólo en el estado seleccionado

solicitud_contin        ::= "+" ESPACIO (texto_respuesta /
                            base64)

copia   ::= "COPY" ESPACIO conjunto ESPACIO buzón

CR      ::= <ASCII CR, retorno de carro, 0x0D>

crear   ::= "CREATE" ESPACIO buzón
            ;; El uso del INBOX provoca un error NO

CRLG    ::= CR LF

CTL     ::= <cualquier carácter de control ASCII y DEL,
           0x00  - 0x1f, 0x7f>

fecha   ::= texto_fecha / <"> texto_fecha <">

fecha_día       ::= 1*2dígitos
```

```
                    ;; Día del mes

fecha_día_fijo  ::= (ESPACIO dígito) / 2dígitos
                    ;; versión en formato fijo de fecha_día

mes_fecha       ::= "Jan" / "Feb" / "Mar" / "Apr" / "May" /
                    "Jun"/ "Jul" / "Aug" / "Sep" / "Oct" /
```



```
                    "Nov"/ "Dec"

texto_fecha     ::= fecha_día  "-" fecha_mes "-" fecha_año

fecha_año       ::= 4dígitos

fecha_hora      ::= <"> fecha_día_fijo "-" fecha_mes "-"
                    fecha_año
                    ESPACIO hora ESPACIO zona <">

borrar  ::= "DELETE" ESPACIO buzón
            ;; El uso del INBOX devuelve un error NO


dígito  ::= "0" / digito_nz

digito_nz       ::= "1" / "2" / "3" / "4" / "5" / "6" / "7" /
                    "8" / "9"

envoltorio      ::= "(" fecha_envol ESPACIO asunto_envol
                    ESPACIO desde_envol ESPACIO remite_envol
                    resp_a_envol ESPACIO a_envol ESPACIO
                    cc_envol ESPACIO
                   bcc_envol ESPACIO en_resp_a_envol ESPACIO
                    id_mensg_envol ")"

bcc_envol       ::= "(" 1*dirección ")"/ nil

cc_envol        ::= "("2*dirección ")" / nil

fecha_envol     ::= cadenan

desde_envol     ::= "(" 1*dirección ")"/ nil

en_resp_a_envol ::= cadenan

id_mensg_envol  ::= cadenan

resp_a_envol    ::= "("1*dirección ")" / nil

remite_envol    ::= "("1*dirección ")" / nil

asunto_envol    ::= cadenan

a_envol ::= "(" 1*dirección ")" / nil

examinar        ::= "EXAMINE" ESPACIO buzón
```



```
recuperar       ::= "FETCH" ESPACIO conjunto ESPACIO ("ALL" /
                    "FULL" / "FAST" / atrib_recuperar / "("
```

```
                    1#atrib_recuperar ")")

atrib_recuperar ::= "ENVELOPE" / "FLAGS" / "INTERNALDATE" /
                    "RFC822" [".HEADER" / ".SIZE" / ".TEXT"] /
                    "BODY" ["STRUCTURE"] / "UID" /
                    "BODY" [".PEEK"] sección
                    ["<" número "." número_nz ">"]

bandera         ::= "\Answered" / "Flagged" / "\Deleted" /
                    "\Seen" / "\Draft" / clave_bandera /
                    extensión_bandera

extensión_bandera       ::= "\"átomo
                            ;; futura expansión. Las
                            ;; implementaciones de cliente
                            ;; DEBEN aceptar banderas
                            ;; extensión_bandera. Las
                            ;; implementaciones de servidor
                            ;; DEBEN generar banderas
                            ;; extensión_bandera excepto si
                            ;; son definidas en una revisión
                            ;; estándar o en vías de
                            ;; estandarización de esta
                            ;; especificación.

clave_bandera   ::= átomo

lista_banderas  ::= "(" #bandera ")"

reconocimiento  ::= "*" ESPACIO (resp_cond_auth /
                    resp_cond_bye)CRLF

campo_nombre_cabecera   ::= acadena

lista_cabecera  ::= "(" 1#campo_nombre_cabecera ")"

LF      ::= <ASCII LF, nueva línea, 0x0A>

lista   ::= "LIST" ESPACIO buzón ESPACIO lista_buzón

lista_buzón     ::= 1*(ATOMO_CARACTER / lista_comodines) /
                    cadena

lista_comodines ::= "%" / "*"

literal         ::= "{" número "}"CRLF *CARACTER8
```



```
                    ;; Número representa el número de octetos
                   ;; CARACTER8

entrar          ::= "LOGIN" ESPACIO id_usuario ESPACIO
                    contraseña

lsub            ::= "LSUB" ESPACIO buzón ESPACIO lista_buzón

buzón           ::= "INBOX" / acadena
                   ;; INBOX es no sensible a mayúsculas-
                    ;; minúsculas.
                    ;; Todas las variantes escritas de INBOX
                    ;; (p.ej."iNbOx" DEBE ser interpretada
                    ;; como INBOXno como una acadena. Ver
                    ;; sección 5.1 para más información
                    ;; acerca de la semántica de los
                    ;; nombres de buzón.

datos_buzón     ::= "FLAGS" ESPACIO lista_banderas /
```

```
                    "LIST" ESPACIO lista_buzón /
                    "LSUB" ESPACIO lista_buzón /
                    "MAILBOX" ESPACIO texto /
                     "SEARCH" [ESPACIO 1#número_nz] /
                     "STATUS" ESPACIO buzón ESPACIO
                     "("#<atrib_estado número ")" /
                     número ESPACIO "EXISTS" / número
                     ESPACIO "RECENT"

lista_buzón     ::= "(" #("\Marked" / "\Noinferiors" /
                    "\Noselect" / "\Unmarked" /
                    extensión_bandera) ")"ESPACIO
                    (<"> CARACTER_ENT_COMILLAS <"> /
                    nil) ESPACIO buzón

multimedia_básico       ::= (<"> ("APPLICATION" / "AUDIO" /
                            "IMAGE" / "MESSAGE" / "VIDEO")
                            <">) / cadena)
                           ESPACIO subtipo_multimedia
                           ;; definido en [MIME_IMT]

mensg_multimedia        ::= <"> "MESSAGE" <"> ESPACIO <">
                            "RFC822"
                            <">
                            ;; definido en [MIME-IMT]

subtipo_multimedia      ::= cadena
                            ;; definido en [MIME_IMT]
```



```
texto_multimedia        ::= <"> "TEXT" <"> ESPACIO
                            subtipo_multimedia
                            ;; definido en [MIME_IMT]

datos_mensg             ::= número_nz ESPACIO
                             ("EXPUNGE" /
                             ("FETCH" ESPACIO atrib_mensg))

atrib_mensg             ::= "(" 1#("ENVELOPE" ESPACIO
                            envoltorio / "FLAGS" ESPACIO
                            "(" #bandera / "\Recent"
                            ")")" / "INTERNALDATE" ESPACIO
                            fecha_hora / "RFC822" [".HEADER"
                            / ".TEXT"] ESPACIO cadenan /
                            "RFC822.SIZE" ESPACIO número /
                            "BODY" ["STRUCTURE"]ESPACIO
                            cuerpo / "BODY" sección ["<"
                            número ">"] ESPACIO cadenan /
                            "UID" ESPACIO id_único) ")"

nil     ::= "NIL"

cadenan ::= cadena / nil

número  ::= 1*dígito
            ;; entero 32-bit sin signo
          ;; (0 <= n < 4,294,967,296

número_nz       ::= dígito_nz *dígito
                    ;; entero 32-bit sin signo distinto de
                   ;; cero (0 < n < 4,294,967,296

contraseña      ::= acadena

entre_comillas  ::= <"> *CARACTER_ENT_COMILLAS <">
```

```
CARACTER_ENT_COMILLAS   ::= <cualquier CARACTER_TEXTO excepto
                            los espec_entre_comillas> / "\"
                            espec_entre_comillas

espec_entre_comillas    ::= <"> / "\"


renombrar               ::= "RENAME" ESPACIO buzón ESPACIO
                            buzón
                            ;; el uso del INBOX como destino
                            ;; devuelve unerror NO
```



```
respuesta               ::= *(solic_cont / datos_resp)
                            resp_hecha

datos_resp              ::= "*" ESPACIO (estado_cond_resp /
                            resp_cond_bye /datos_buzón /
                            datos_mensg / datos_aptitud)

                            CRLF

resp_hecha              ::= resp_etiq / resp_fatal

resp_fatal              ::= "*" ESPACIO resp_cond_bye CRLF
                           ;; El servidor cierra la
                            ;; conexión de inmediato

resp_etiq               ::= etiqueta ESPACIO resp_cond_estado
                            CRLF

resp_cond_auth  ::= ("OK" / "PREAUTH") ESPACIO text_resp
                   ;; condición de autentificación

resp_cond_bye   ::= "BYE" ESPACIO texto_resp

resp_cond_estado        ::= ("OK" / "NO" / "BAD") ESPACIO
                            texto_resp
                            ;; condición de estado

texto_resp              ::= ["[" código_texto_resp "]" ESPACIO]
                           (texto_mime2 / texto)
                            ;; el texto NO DEBERIA comenzar con
                            ;; "[" o "="

código_texto_resp       ::= "ALERT" / "PARSE" /
                            "PERMANENTFLAGS" ESPACIO "("
                            #(bandera /
                            "\*")")" /
                            "READ-ONLY" / "READ-WRITE" /
                            "TRYCREATE"
                            / "UIDVALIDITY" ESPACIO número_nz /
                            "UNSEEN" ESPACIO número_nz /
                            átomo [ESPACIO 1*<cualquier
                            CARACTER_TEXTO excepto "]">]

buscar          ::= "SEARCH" ESPACIO ["CHARSET" ESPACIO acadena
                    ESPACIO] 1#clave_búsqueda
                   ;; [CHARSET] DEBE estar registrado en la
                    ;: IANA
```

```
clave_búsqueda  ::= "ALL" / "ANSWERED" / "BCC" ESPACIO acadena
                    /"BEFORE" ESPACIO fecha / "BODY" ESPACIO
                    acadena /
                    "CC" ESPACIO acadena / "DELETED" /
                    "FLAGGED" / "FROM" ESPACIO acadena /
                    "KEYWORD" ESPACIO clave_bandera / "NEW" /
                    "OLD" /
                    "ON" ESPACIO fecha / "RECENT" / "SEEN" /
                    "SINCE" ESPACIO fecha / "SUBJECT" ESPACIO
                    acadena /
                    "TEXT" ESPACIO acadena / "TO" ESPACIO
                    cadena /
                    "UNANSWERED" / "UNDELETED" / "UNFLAGGED" /
                    "UNKEYWORD" ESPACIO clave_bandera /
                    "UNSEEN" /
                   ;; Todo lo anterior estaba en [IMAP2]
                    "DRAFT" / "HEADER" ESPACIO
                    campo_nombre_cabecera ESPACIO acadena /
                    "LARGER" ESPACIO número / "NOT" ESPACIO
                    clave_búsqueda /
                    "OR" ESPACIO clave_búsqueda ESPACIO
                    clave_búsqueda /
                    "SENTBEFORE" ESPACIO fecha / "SENTON"
                    ESPACIO fecha /
                    "SENTSINCE" ESPACE date / "SMALLER"
                    ESPACIO número /
                    "UID" ESPACIO conjunto / "UNDRAFT" /
                    conjunto
                    / "(" 1#clave_búsqueda ")"

sección ::= "[" [texto_secc / (número_nz *["." número_nz]
           ["." (texto_secc / "MIME")])] "]"

texto_secc      ::= "HEADER" / "HEADER.FIELDS" [".NOT"]
                    ESPACIO
                    lista_cabecera / "TEXT"

seleccionar     ::= "SELECT" ESPACIO buzón

número_sec      ::= número_nz / "*"
                ;; * es el mayor número en uso. Para los
                ;; números de secuencia de mensaje, es
                ;; el número de mensajes contenidos en el
                ;; buzón. Para los identificadores únicos,
                ;; se trata del identificador único del
                ;; último mensaje del buzón.

conjunto        ::= núm_secuencia / (núm_secuencia ":"
```



```
                    núm_secuencia) / (conjunto "," conjunto)
                   ;; Identifica un conjunto de mensajes.
                    ;; En el caso de los números de secuencia,
                    ;; estos son números consecutivos
                    ;; partiendo del 1 al número total de
                    ;; mensajes en el buzón. El carácter
                    ;; coma se usa para delimitar números
                    ;; individuales, los dos puntos delimitan
                    ;; el intervalo entre dos números ambos
                    ;; inclusive. Ejemplo: 2,4:7,9,12:* es
                   ;; 2,4,5,6,7,9,12,13,14,15 para un buzón
                    ;; de 15 mensajes.
```

```
ESPACIO         ::= <ASCII SP, espacio, 0x20>

estado  ::= "STATUS" ESPACIO buzón ESPACIO "(" 1#atrib_estado
            ")"

atrib_estado ::= "MESSAGES" / "RECENT" / "UIDNEXT" /
                 "UIDVALIDITY" / "UNSEEN"

almacenar ::= "STORE" ESPACIO conjunto ESPACIO
              atrib_banderas_almac

atrib_banderas_almac ::= (["+" / "-"] "FLAGS" [".SILENT"])
                         ESPACIO
                        (lista_banderas / #bandera)

cadena ::= entre_comillas / literal

suscribir ::= "SUBSCRIBE" ESPACIO buzón

etiqueta ::= 1*<cualquier ATOMO_CARACTER excepto "+">

texto ::= 1*TEXT_CHAR

texto_mime2 ::= "=?" <conjunto_de_caracteres> "?"
                <codificación> "?"
               <texto_codificado> "?="
                ;; Sintaxis definida en [MIME-HDRS]

CARACTER_TEXTO ::= <cualquier carácter excepto CR y LF>

hora ::= 2dígitos ":" 2dígitos ":" 2dígitos
         ;; Horas minutos segundos
```



```
uid  ::= "UID" ESPACIO (copiar / recuperar / buscar /
         almacenar)
         ;; Se utilizan identificadores únicos en lugar
         ;; de números de secuencia de mensaje

id_único        ::= número_nz
                    ;; orden ascendente

abandonar_suscripción ::= "UNSUBSCRIBE" ESPACIO buzón

id_usuario      ::= acadena

comando_x       ::= "X" átomo <argumentos de comando
                    experimental>

zona            ::= ("+" / "-") 4digitos
                    ;; Cuatro dígitos con signo
                    ;; representando elvalor de hhmm
                    ;; horas y minutos al oeste de
                    ;; Greenwich (es decir, (el error
                    ;; asociado a la hora dada con
                    ;; respecto a la horaUniversal).
                    ;; Si restamos a la hora dada la
                    ;; hora propia de la zona obtendremos
                    ;; la hora en forma UT. La zona
                    ;; horaria Universal es"+0000".
```

10. Notas del autor

Este documento es una revisión o reescritura de documentos anteriores, y sustituye la especificación del protocolo

contenida en los documentos: RFC 1730, el documento IMAP2bis.TXT no publicado, RFC 1176, y RFC 1064.

11. Consideraciones de seguridad

Las transacciones en el protocolo IMAP4rev1, incluyendo los datos de correo electrónico, son enviadas a la red sin preocuparnos por una posible intromisión en los mismos, a menos que se acuerde un mecanismo de protección de la privacidad en el comando AUTHENTICATE.

Un mensaje de error del servidor para un comando AUTHENTICATE fallido debido a una entrega de credenciales no válidas NO DEBERIA ofrecer detalles acerca del motivo por el cual estas son no válidas.

El uso del comando LOGIN envía las claves sin encriptación.

Esto puede remediarse si utilizamos en su lugar el comando AUTHENTICATE.

Un mensaje de error de servidor para un comando LOGIN fallido, NO DEBERIA especificar que el nombre de usuario, al contrario que la contraseña, es no válido.

Para conocer más acerca de consideraciones de seguridad se podrán encontrar en la sección donde se tratan los comandos AUTHENTICATE y LOGIN.

12. Dirección del Autor

Mark R. Crispin
Networks and Distributed Computing
University of Washington
4545 15th Aveneue NE
Seattle, WA 98105-4527
Teléfono: (206) 543-5762
Correo electrónico: MRC@CAC.Washington.EDU ..

Apéndices

A Referencias

[ACAP] Myers, J. "ACAP -- Application Configuration Access Protocol", Work in Progress.

[CHARSET] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1700, USC/Information Sciences Institute, October 1994.

[DISPOSITION] Troost, R., and Dorner, S., "Communicating Presentation Information in Internet Messages: The Content-Disposition Header", RFC 1806, June 1995.

[IMAP-AUTH] Myers, J., "IMAP4 Authentication Mechanism", RFC 1731. Carnegie-Mellon University, December 1994.

[IMAP-COMPAT] Crispin, M., "IMAP4 Compatibility with IMAP2bis", RFC 2061, University of Washington, November 1996.

[IMAP-DISC] Austein, R., "Synchronization Operations for Disconnected IMAP4 Clients", Work in Progress.

[IMAP-HISTORICAL] Crispin, M. "IMAP4 Compatibility with IMAP2 and IMAP2bis", RFC 1732, University of Washington, December 1994.

[IMAP-MODEL] Crispin, M., "Distributed Electronic Mail Models in

IMAP4", RFC 1733, University of Washington, December 1994.

[IMAP-OBSOLETE] Crispin, M., "Internet Message Access Protocol - Obsolete Syntax", RFC 2062, University of Washington, November 1996.

[IMAP2] Crispin, M., "Interactive Mail Access Protocol - Version 2", RFC 1176, University of Washington, August 1990.

[LANGUAGE-TAGS] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[MD5] Myers, J., and M. Rose, "The Content-MD5 Header Field", RFC 1864, October 1995.

[MIME-IMB] Freed, N., and N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[MIME-IMT] Freed, N., and N. Borenstein, "MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types", RFC 2046, November 1996.

[MIME-HDRS] Moore, K., "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC-822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, University of Delaware, August 1982.

[SMTP] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, USC/Information Sciences Institute, August 1982.

[UTF-7] Goldsmith, D., and Davis, M., "UTF-7: A Mail-Safe Transformation Format of Unicode", RFC 1642, July 1994.

B. Variaciones respecto a RFC 1730

1) Se ha añadido el comando STATUS.

2) Establecer como sintaxis formal que el constructor "#" nunca puede hacer referencia a múltiples espacios.

3) Se ha trasladado la sintaxis obsoleta a un documento independiente.

4)El comando PARTIAL ha quedado obsoleto.

5)Los atributos de recuperación RFC822.HEADER.LINES,

RFC.HEADER.LINES.NOT, RFC822.PEEK, y RFC822.TEXT.PEEK han quedado obsoletos.

6)Se han añadido los sufijos "<" origen "." tamaño ">" para los atributos de texto del BODY.

7)Los especificadores de parte HEADER, HEADER.FIELDS, HEADERS.FIELDS.NOT, MIME y TEXT han sido añadidos.

8) Se ha añadido soporte para disposición de contenido y lenguaje de contenido.

9) Se ha eliminado la restricción de recuperación de partes MULTIPART anidadas.

10) El número de parte de cuerpo 0 ha quedado obsoleto.

11) Los autentificadores soportados por servidor se identifican ahora como aptitudes.

12) La funcionalidad que identifica este protocolo se denomina ahora "IMAP4rev1". Un servidor que ofrezca compatibilidad hacia atrás para RFC 1730 DEBERIA emitir la aptitud "IMAP4" además de la aptitud "IMAP4rev1" en su respuesta CAPABILITY.

13) Se ha añadido una descripción de la convención utilizada en el espacio de nombres para nombres de buzón.

14) Se ha añadido una descripción de la convención utilizada en los nombres de buzón internacionales.

15) Los elementos de estado UID-NEXT y UID-VALIDITY se denominan ahora UIDNEXT y UIDVALIDITY. Esto representa un cambio con respecto a IMAPSTATUS Work in Progress pero no respecto a RFC 1730.

16) Se añade una aclaración, un argumento de nombre de buzón null en el comando LIST devuelve una respuesta no etiquetada LIST incluyendo el delimitador de jerarquía y raíz del argumento de referencia.

17) Se definen términos tales como "DEBE","DEBERIA", y "NO DEBE".

18) Se añade una sección que define los atributos de mensaje y más específicamente detalles de la semántica de los números de secuencia de mensaje, UIDS y banderas.

19) Explica con detalle aquellas circunstancias en las que un cliente puede enviar múltiples comandos sin tener que esperar a la respuesta, y aquellas circunstancias en las cuales puede surgir ambigüedad.

20) Añade recomendaciones acerca del comportamiento del servidor en relación a los comandos DELETE y RENAME cuando existen nombres de jerarquía inferiores para el nombre dado.

21) Indica que un nombre de buzón no puede ser eliminado de manera unilateral por el servidor, incluso si dicho nombre de buzón no existe.22) Indica que LIST debería devolver su resultado con celeridad sin ningún retraso injustificado.

23) Indica que el argumento fecha_hora del comando APPEND establece la fecha interna del mensaje.

24) Indica la manera de actuar del comando APPEND cuando el buzón destino es el buzón actualmente seleccionado.

25) Indica que los cambios producidos en las banderas por agentes externos deberían ser comunicados vía FETCH no etiquetado incluso si el comando actual se trata de un comando almacenar con el sufijo ".SILENT".

26) Indica que el comando COPY añade al buzón destino.

27) Añade el código de respuesta NEWNAME

28) Reescribe la descripción de la respuesta no etiquetada BYE para clarificar su semántica.

29) Cambia la referencia para el MD5 del cuerpo para hacer referencia al RFC apropiado.

30) Deja claro que la sintaxis formal contiene reglas que pueden solaparse, y en el caso en que esto suceda, la regla que tome lugar en primer lugar tendrá preferencia sobre cualquier regla posterior.

31) Corrige la definición de campo_param_cuerpo

32) Más sintaxis formal para los datos de funcionalidad.

33) Establecer que cualquier variación caligráfica de "INBOX" debe interpretarse como INBOX.

34) Establecer que el texto inteligible por el usuario en

texto_resp no debería comenzar con "[" o "=".

35) Cambiar las referencias MIME a los documentos Draft Standard.

36) Establecer claramente la semántica de \Recent

37) Ejemplos adicionales

C. Glosario